SPAWAR
Systems Center
San Diego

TECHNICAL REPORT 1812
December 1999

# Multimode Acoustic Source

Newell O. Booth
Shelby F. Sullivan, Sr.

Approved for public release;
distribution is unlimited.

SSC San Diego

TECHNICAL REPORT 1812
December 1999

# Multimode Acoustic Source

Newell O. Booth
Shelby F. Sullivan, Sr.

Approved for public release;
distribution is unlimited.

SPAWAR
Systems Center
San Diego

SSC San Diego
San Diego, CA 92152–5001

SSC SAN DIEGO
San Diego, California 92152–5001

E. L. Valdes, CAPT, USN
Commanding Officer

R. C. Kolb
Executive Director

## ADMINISTRATIVE INFORMATION

This work, conducted during FY 1993, was sponsored in part by the following organizations:

Naval Air Warfare Center, Airborne Active Adjunct Project, N2269/93/WX/00199
Naval Air Systems Command, AIR–546W8, Advanced Collection Technology,
A5465461/003C/3H20890001
NCCOSC, RDT&E Division, Marine Sciences and Technology Department,
Major Bid and Proposal Funds

Released by
Dr. C. D. Rees, Head
Acoustic Branch

Under authority of
C. J. Sayre, Head
Electromagnetics &
Advanced Technology
Division

LH

# EXECUTIVE SUMMARY

## OBJECTIVE

Develop the design criteria for an explosive multimode acoustic source predicting the beam-frequency spectra and sidelobe characteristics for each of the basic modes. The source can be programmed remotely (as from an aircraft) to ping in different modes at selectable depths, each mode having a different combination of spectra and beampattern.

## APPROACH

The choice of modes is achieved by controlling the timing and firing order of explosive elements positioned in the array. Mode frequencies, beamwidths, and associated bandwidths are determined when the array is being manufactured by the number of elements and the spacing. Each of the modes has different beam and spectral characteristics which are useful in different environments and in different tactical situations. The acoustic characteristics of the source in each of the modes are described and acoustic design criteria are derived. Application examples are illustrated.

## RESULTS

The analysis predicts that a multimode acoustic source can be achieved with sidelobe levels <–25 dB below the main lobes in all modes with 10% tolerances in timing, spacing, and the weight of explosive material. Such a source has the operational flexibility to choose the beam and spectral characteristics which optimize the active system performance in a given acoustic environment while having but one item in inventory. Predictions contained in this report illustrate significant echo-to-interference-ratio improvements that are achievable over single mode or omnidirectional sources.

## RECOMMENDATIONS

1. Develop and test low-cost electronic timing and firing circuitry.
2. Analyze and test the performance of the various modes in various ocean environments, developing source–receiver system deployment strategies.
3. Develop signal processing and operator interface software and specifications for implementation in military computers.

# CONTENTS

## Figures

# DESCRIPTION OF MULTIMODE ACOUSTIC SOURCE

The Multimode Acoustic Source is a vertical array of chemical charges which are fired in a programmable sequence to achieve the acoustic beam and spectral characteristic which is most suitable for a particular mission or ocean environment.

The components of the Multimode Source in a sonobuoy configuration are shown in figure 1. A package of several source arrays deployed on the surface of the ocean is shown on the left. The package has a radio receiver which receives remote commands and relays them to each of the source arrays. The source array, shown in the center before deployment and on the right as deployed at the time of the ping, consists of a number of chemical charges spaced a distance, *d*, apart. The chemical charges contain either detonating or deflagrating materials which generate acoustic impulses.

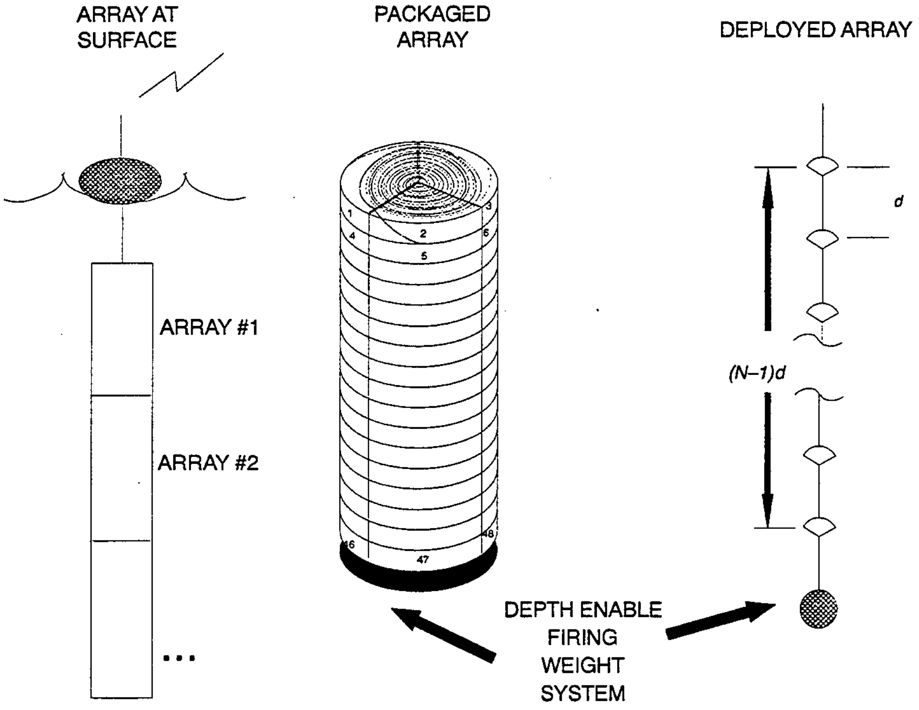

**Figure 1.** Multimode source components.

Each charge contains an electronic initiator (EI). An example of one implementation is diagrammed in figure 2. The initiator consists of a DC-to-DC power converter to charge a high-voltage capacitor which stores energy to fire the ignitor, a timer to control the time delay of firing, a firing-mode detector to set the time delay, a trigger detector to start the timer, and a switch to dump the high-voltage capacitor energy into an ignitor which launches the reaction in the chemical charge. A low-voltage capacitor is included to power the timer after the trigger if the power is cut off by the firing of previous elements. The diode prevents the capacitor from shorting when the power is cut off. The choke prevents the low-voltage capacitor and the DC-to-DC converter from shorting the trigger and mode signals. A resistor across the high-voltage capacitor drains the charge so that if the element fails to fire, the capacitor will discharge, rendering the element safe to recover.

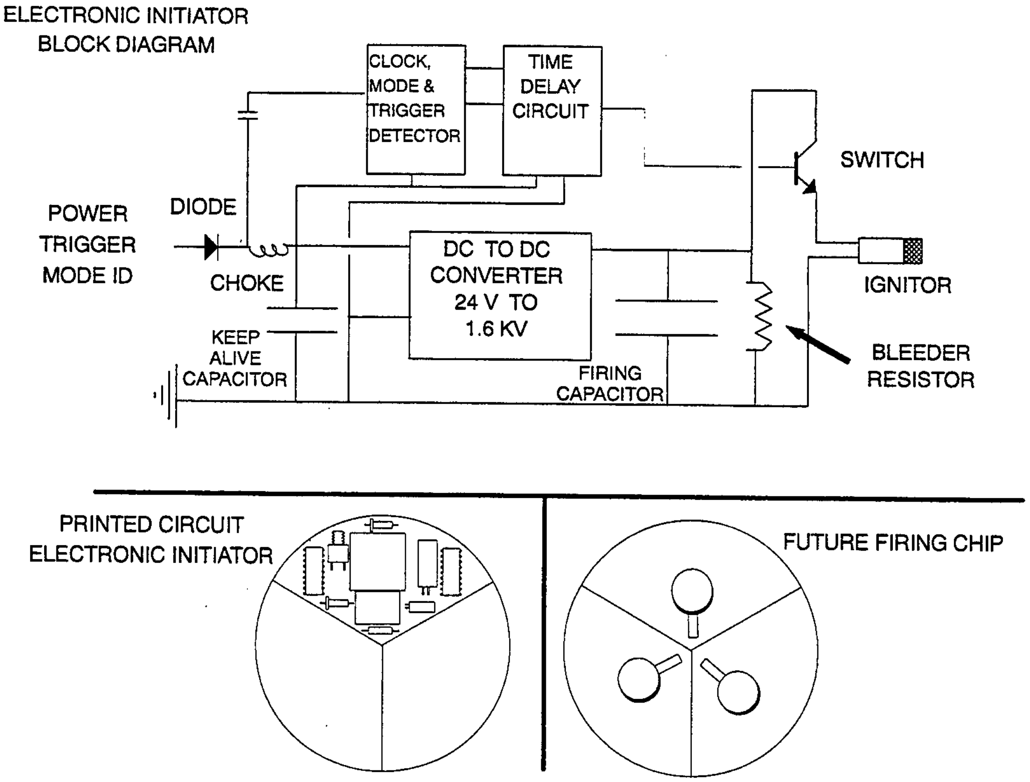

**Figure 2.** Element electronic initiator.

Figure 2 also shows the elements as they might be packaged with printed circuit and special-purpose integrated circuit electronics. The printed-circuit version would be used for testing. The integrated-circuit version would be used in production to reduce the cost and size of the electronic initiator. The chemical charges would be packaged to fill the voids around the initiator electronics.

The depth enable and firing unit (DEFU) is shown in figure 3. It consists of a seawater battery that also functions as a weight to deploy the array after release. The mechanical safing and arming switch in each DEFU is closed before and during deployment. When the surface unit is deployed, the seawater batteries are activated, powering the electronics in the DEFU. The DEFU of the next array to be deployed receives a command signal via a radio receiver in the surface unit. The command signal contains a mode identification code and a firing depth, which are stored in the DEFU. A second command signal releases the array.

When the array passes through an arming depth, power is supplied to the element electronics, charging the firing capacitor and allowing the elements to receive the mode identification signal, which has been stored in the DEFU. When the depth sensor passes the firing depth, the DEFU sends a trigger signal to the elements, starting the timing sequence.

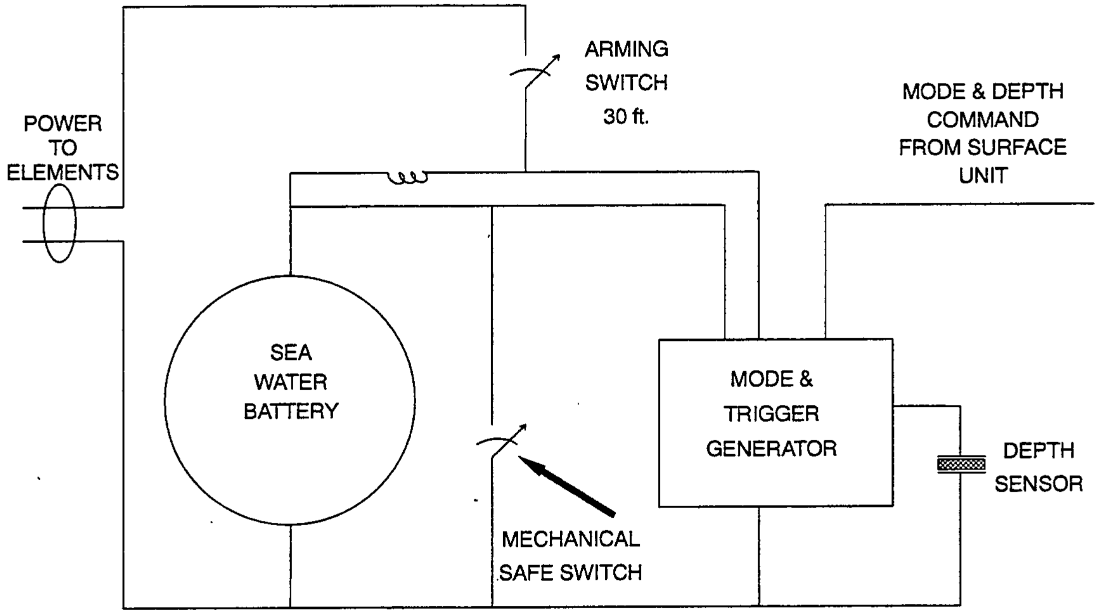

**Figure 3**. Depth enable and firing unit.

Other partitioning of the functions between the DEFU and the EI is possible. In particular, the EI can be simplified by moving the DC-to-DC converter to the DEFU and accepting the reliability risk of running high voltage between the DEFU and the elements.

# ACOUSTIC CHARACTERISTICS

## BEAM-FREQUENCY SPECTRUM CALCULATION

By choosing the relative timing of the impulses, several different beam patterns and source spectra can be obtained. This section derives formulae for calculating the beam patterns and source levels for shaded multimode source arrays.

Consider a line array of $N$ acoustic sources at positions $d_n$, as illustrated in figure 4. Each source generates a pressure signal, $p_n(t)$. The signal in the far field at angle $\theta$ is given by

$$p(t) = \sum_{n=1}^{N} \frac{pn\left(t - t_n - \frac{r - d_n \sin\theta}{c}\right)}{r} \qquad (1)$$

where $t_n$ is the time delay of the $n^{\text{th}}$ signal, $\theta$ is the angle from broadside, $c$ is the speed of sound, and $r$ is the range.

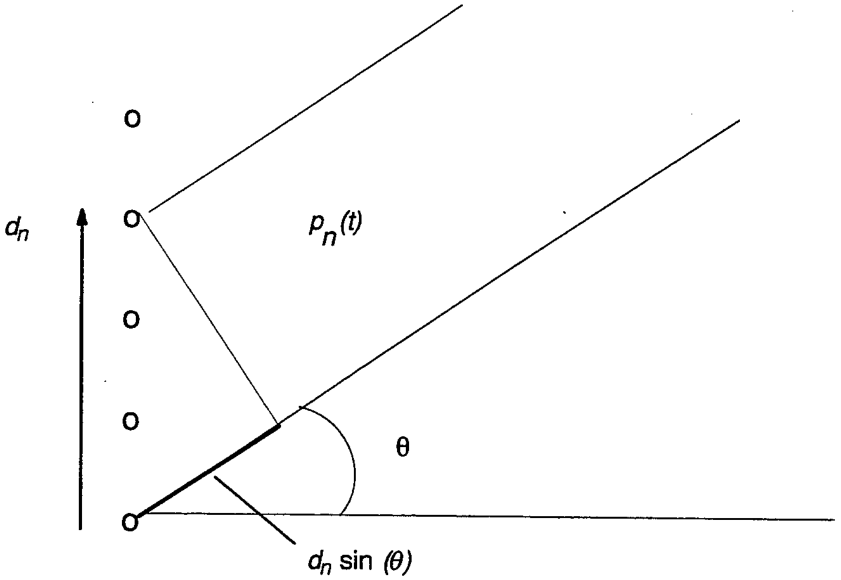

**Figure 4**. Line array of acoustic sources.

For a multimode source with uniform spacing, $d_n$ and $t_n$ are chosen according to

$$d_n = nd \qquad (2)$$

and

$$t_n = n\tau, \quad \tau = \frac{d}{c'} \qquad (3)$$

where $d$ is the element spacing, τ is the time between pulses, and $c'$ is the speed at which the pulses move along the array. We define the Mach number, $M$, as the ratio

$$M = \frac{c'}{c} \qquad (4)$$

where $d$ is the element spacing, $\tau$ is the time between pulses, and $c'$ is the speed at which the pulses move along the array. We define the Mach number, $M$, as the ratio

$$M = \frac{c'}{c} \tag{4}$$

The pressure at range $r$ is then given by

$$p(t) = \sum_{n=1}^{N} \frac{p_n\left(t - \frac{nd}{Mc}\,(1 - M\sin\theta) - \frac{r}{c}\right)}{r} \tag{5}$$

Fourier transforming and suppressing the range dependence, we obtain the Fourier component of the pressure at angle $\theta$ and frequency $f$:

$$p(\,f\,) = \sum_{n=1}^{N} p_n(f)\, e^{-j2\pi f\frac{nd}{Mc}(1-M\sin\theta)} \tag{6}$$

The beam-frequency spectrum is defined as the power in a 1-Hz band leaving the source at frequency $f$ and angle $\theta$ normalized by the power on beam axis. Since the power received at frequency $f$ and angle $\theta$ is proportional to the pressure squared, the beam-frequency spectrum is given by

$$B(f,\theta) \equiv \frac{p(f)\cdot p^{*}(f)}{\left|\sum_{n=1}^{N} p_n(f)\right|^2} \tag{7}$$

For the purpose of illustrating beam-frequency spectra, we assume that all the power signals are identical except for an amplitude shading coefficient, $\alpha_n$.

$$p_n(f) = p_e(f)\alpha_n \tag{8}$$

The amplitude shading coefficients, $\alpha_n$, are chosen such that

$$\sum_{n}^{N} \alpha_n = N \tag{9}$$

With this assumption, the pressure transform becomes

$$p(f) = p_e(f) \sum_{n}^{N} \alpha_n\, e^{-j2\pi f\frac{nd}{Mc}(1-M\sin\theta)} \tag{10}$$

Using Eq. 10, the beam-frequency spectrum is given by

$$B(f,\theta) = \frac{\left|\sum_{n}^{N} \alpha_n\, e^{j2\pi f\frac{nd}{Mc}(1-M\sin\theta)}\right|^2}{N^2}. \tag{11}$$

All the element signals add in phase when $2\pi f\frac{nd}{Mc}(1 - M\sin\theta_s) = 2\pi n$, giving a peak response on beam axis of

$$p_b(f) = \sum_{n=1}^{N} p_n(f) \tag{12}$$

## BEAM-FREQUENCY SPECTRA BY MODE

Equation 11 is used to generate beam-frequency spectra for a 30.2-meter-long 48-element array for each of several modes. The shading coefficients, $\alpha_n$, are shown in figure 5. The shading is a step approximation to Dolph–Chebyshev shading (shown as the smooth curve) used to simplify construction while maintaining sidelobes $\approx -30$ dB.

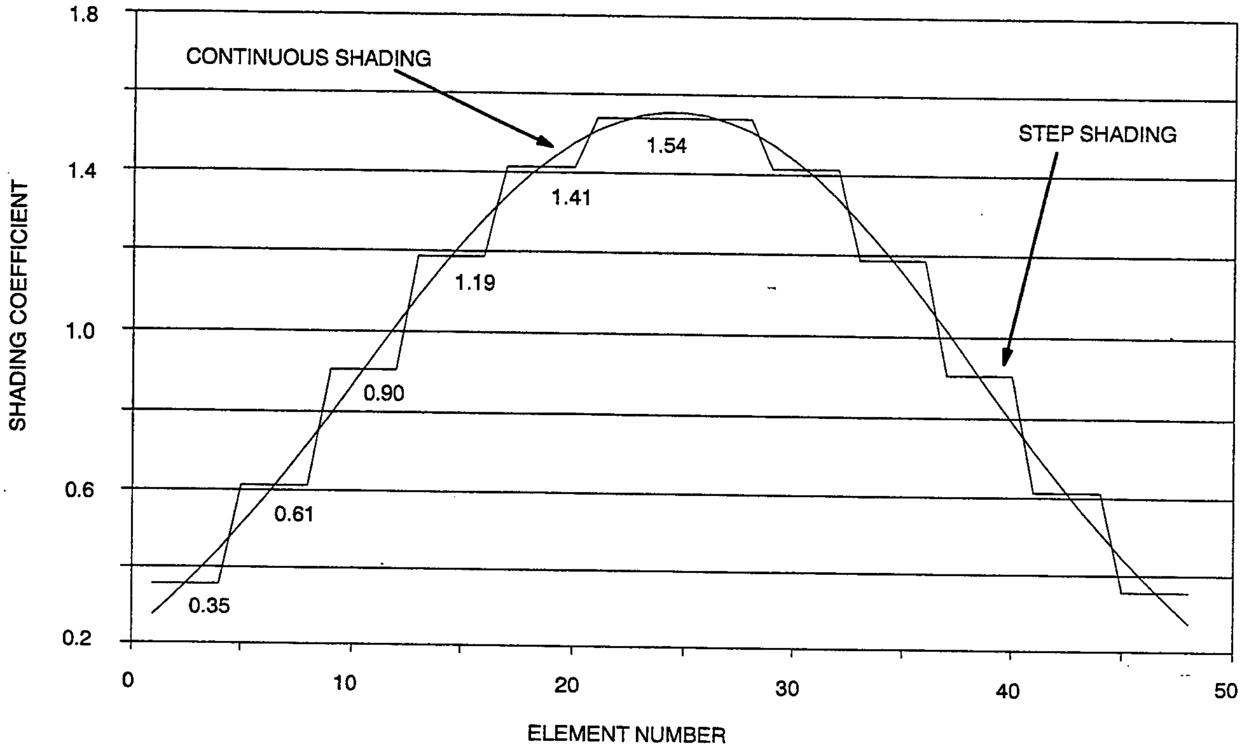

**Figure 5.** Multimode source shading coefficients.

Beam-frequency spectra are shown in figures 6–13 for various modes and Mach numbers. Each figure shows a snapshot of impulse wave fronts from five sources, illustrating the wave front generated in the given mode. Each figure also contains a three-dimensional plot of the beam-frequency spectrum vs frequency and angle. Spectral and beam pattern cuts are shown at labeled frequencies and angles in the lower left and right portions of the figure. The beam patterns at what would be received using a 1-Hz filter on the receiver vs angle. The spectra are the spectra obtained from a broadband receiver at the given angle.

Figure 6 illustrates the beam-frequency spectrum of an array in the broadside mode, where all the elements are fired at once, equivalent to $M = \infty$. As shown in the top left, the wave fronts superimpose to form a single, diffraction-limited wave front at $\alpha = 0$. The three-dimensional plot of the beam-frequency spectrum shows a broadside beam at all frequencies. The spectrum at beam center, shown in the lower left, is flat, consistent with a single diffraction-limited impulse propagating broadside. As

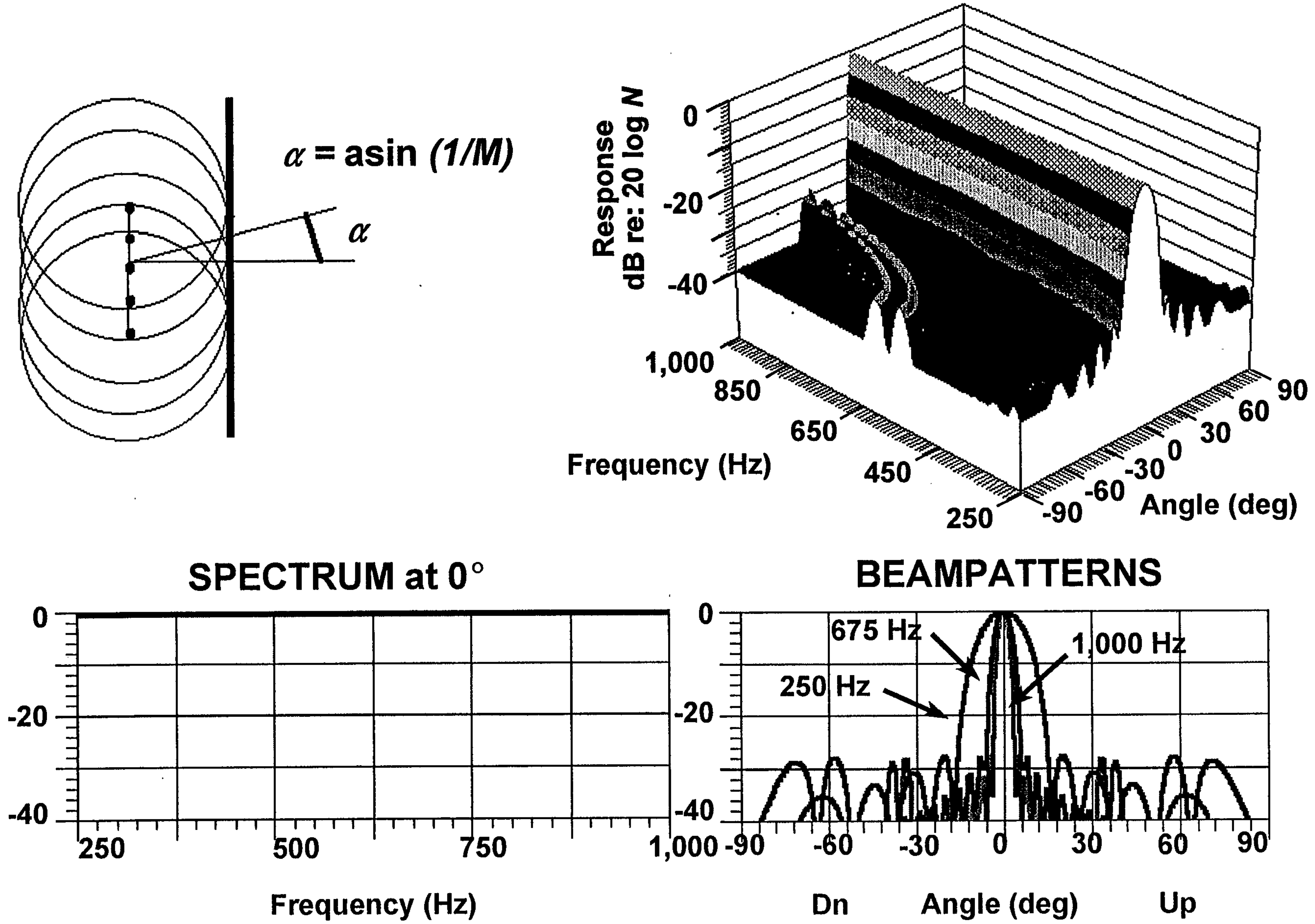

**Figure 6.** Beam-frequency spectrum, broadside mode, $M = \infty$, shaded.

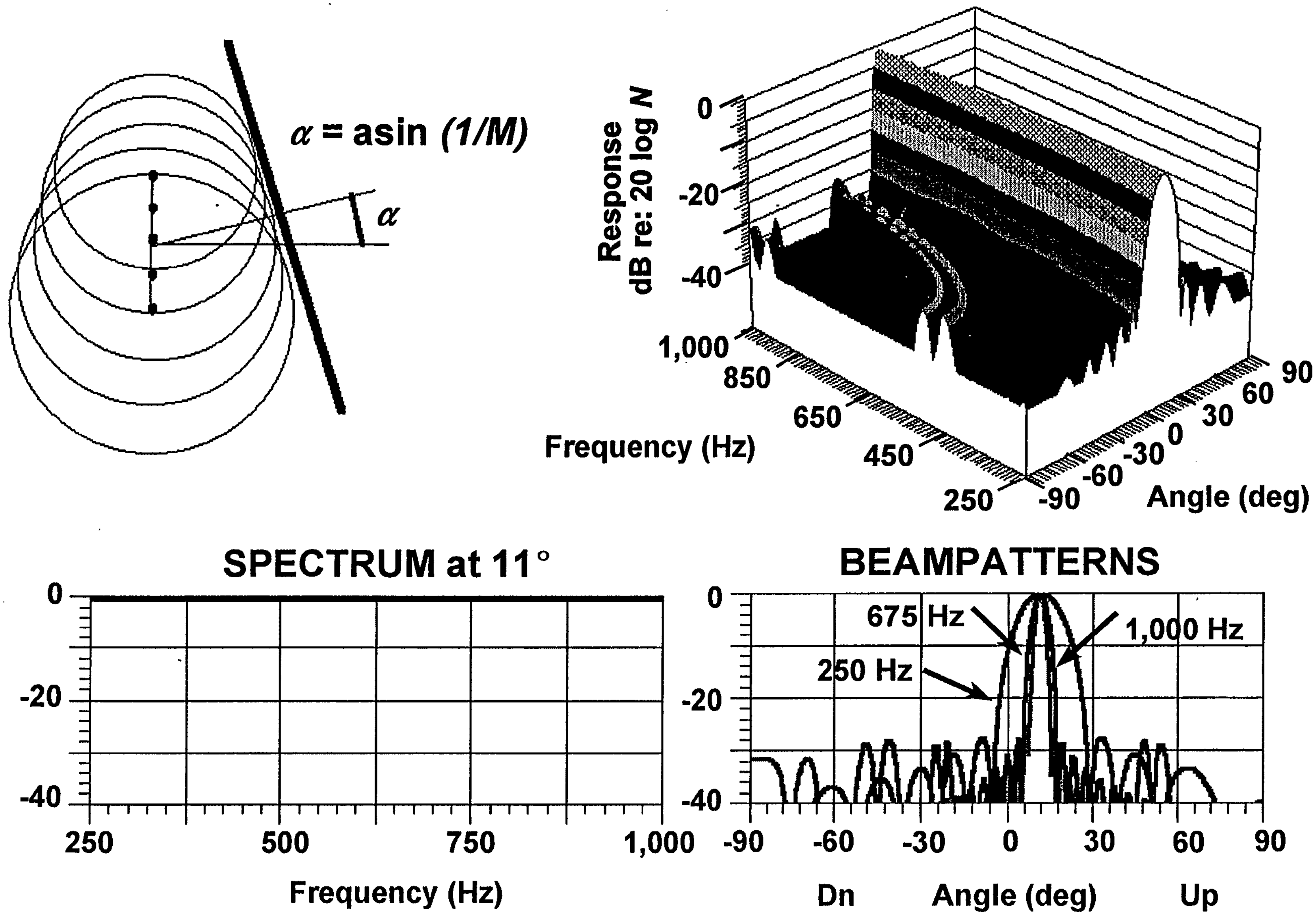

**Figure 7.** Beam-frequency spectrum, end-fired mode, $M = 5$, shaded.

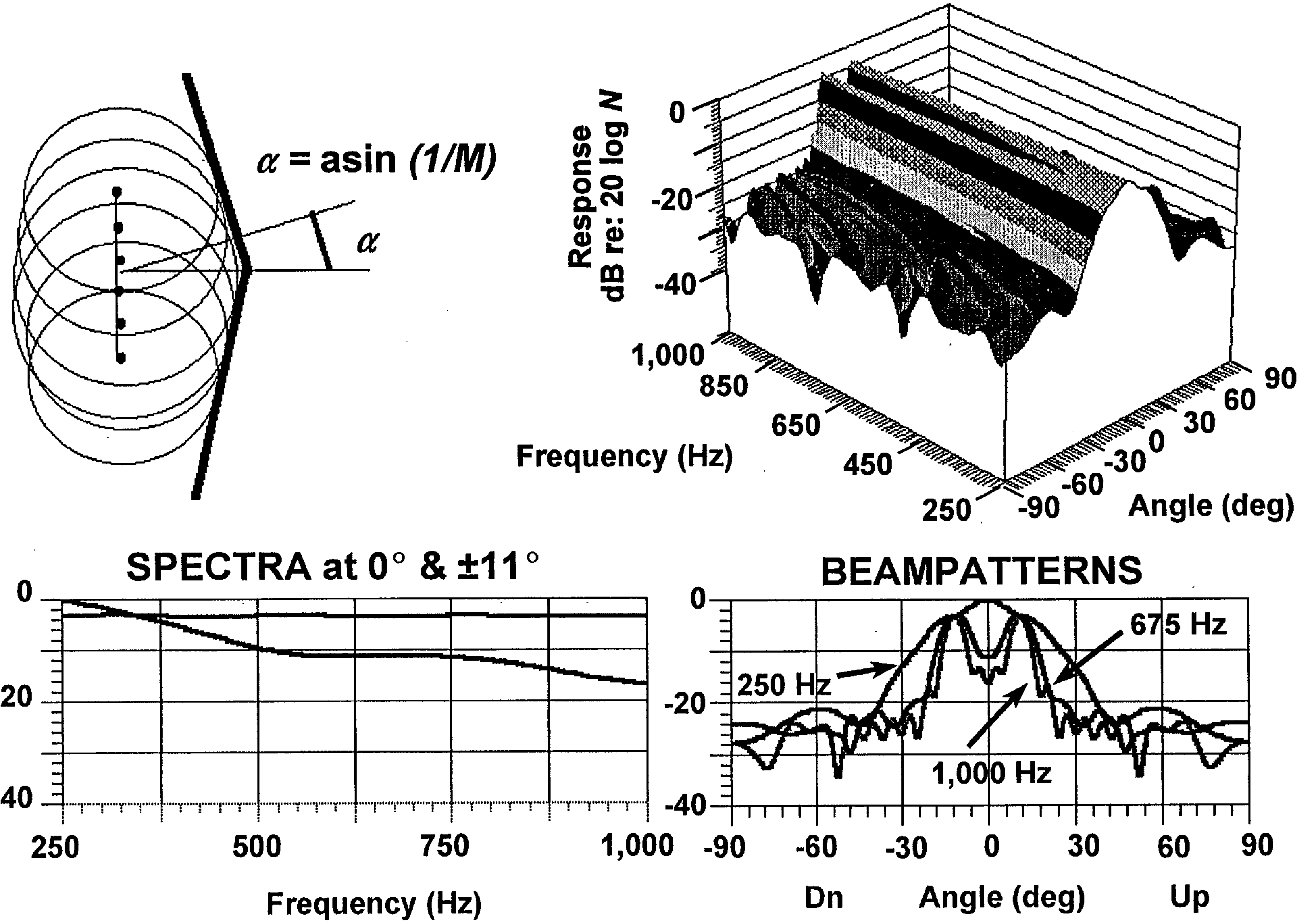

**Figure 8.** Beam-frequency spectrum, center-fired mode,$M$ = 5, shaded.

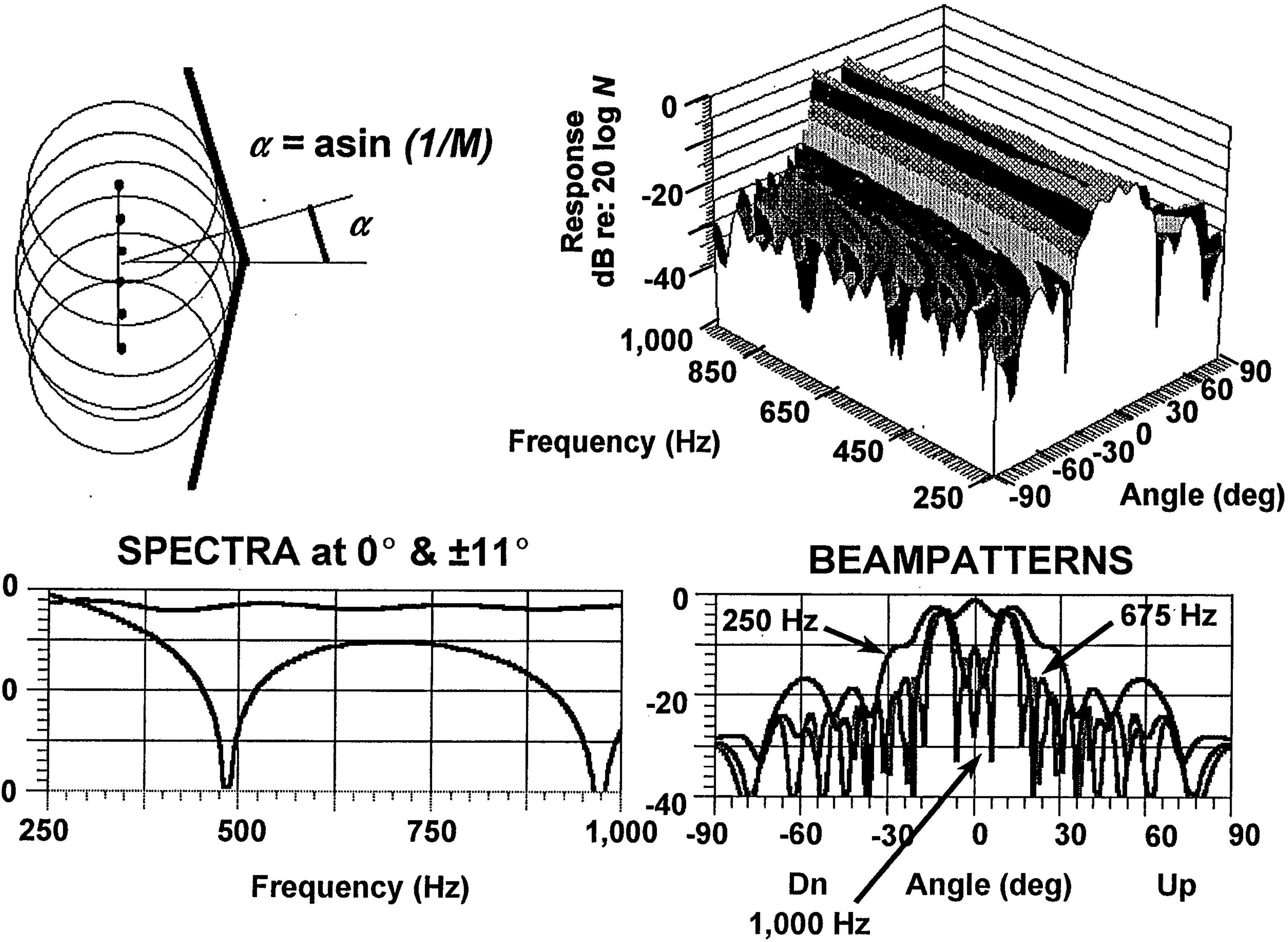

**Figure 9.** Beam-frequency spectrum, center-fired mode, $M = 5$, unshaded.

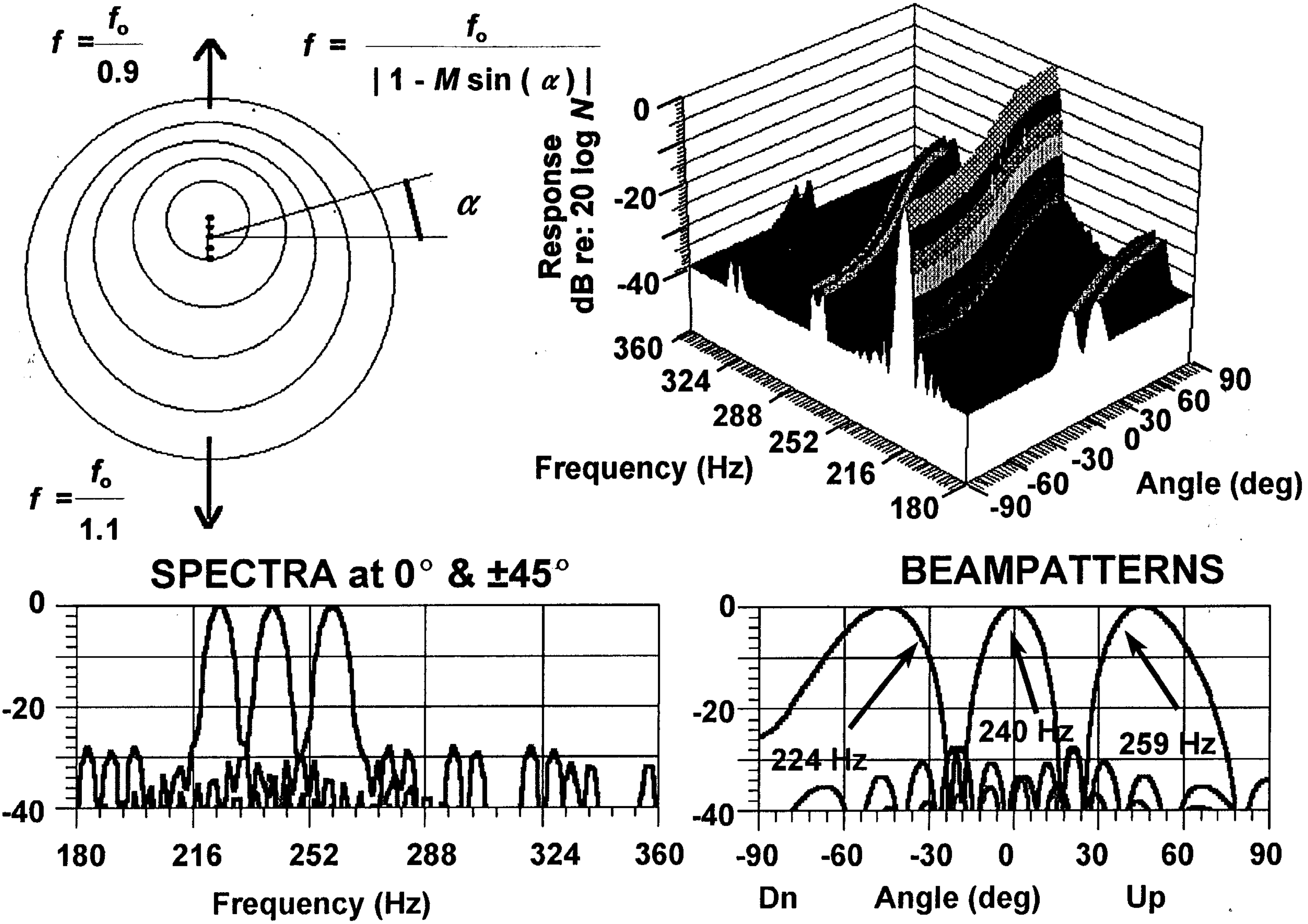

**Figure 10.** Beam-frequency spectrum, SFS mode, $M = 0.1$, $f_o = 240$ Hz, shaded.

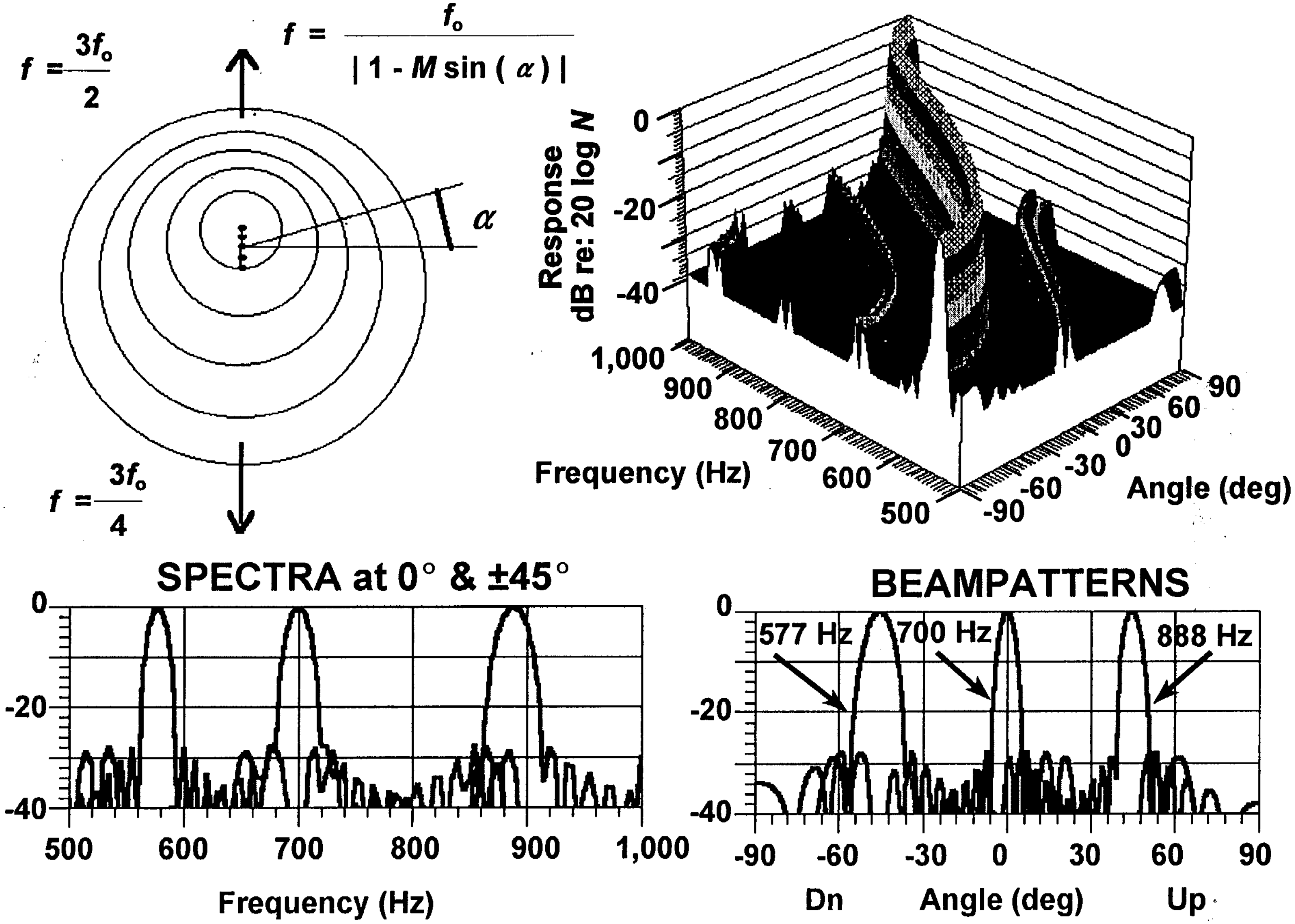

**Figure 11.** Beam-frequency spectrum, SFS mode, $M = 0.3$, $f_o = 700$ Hz, shaded.

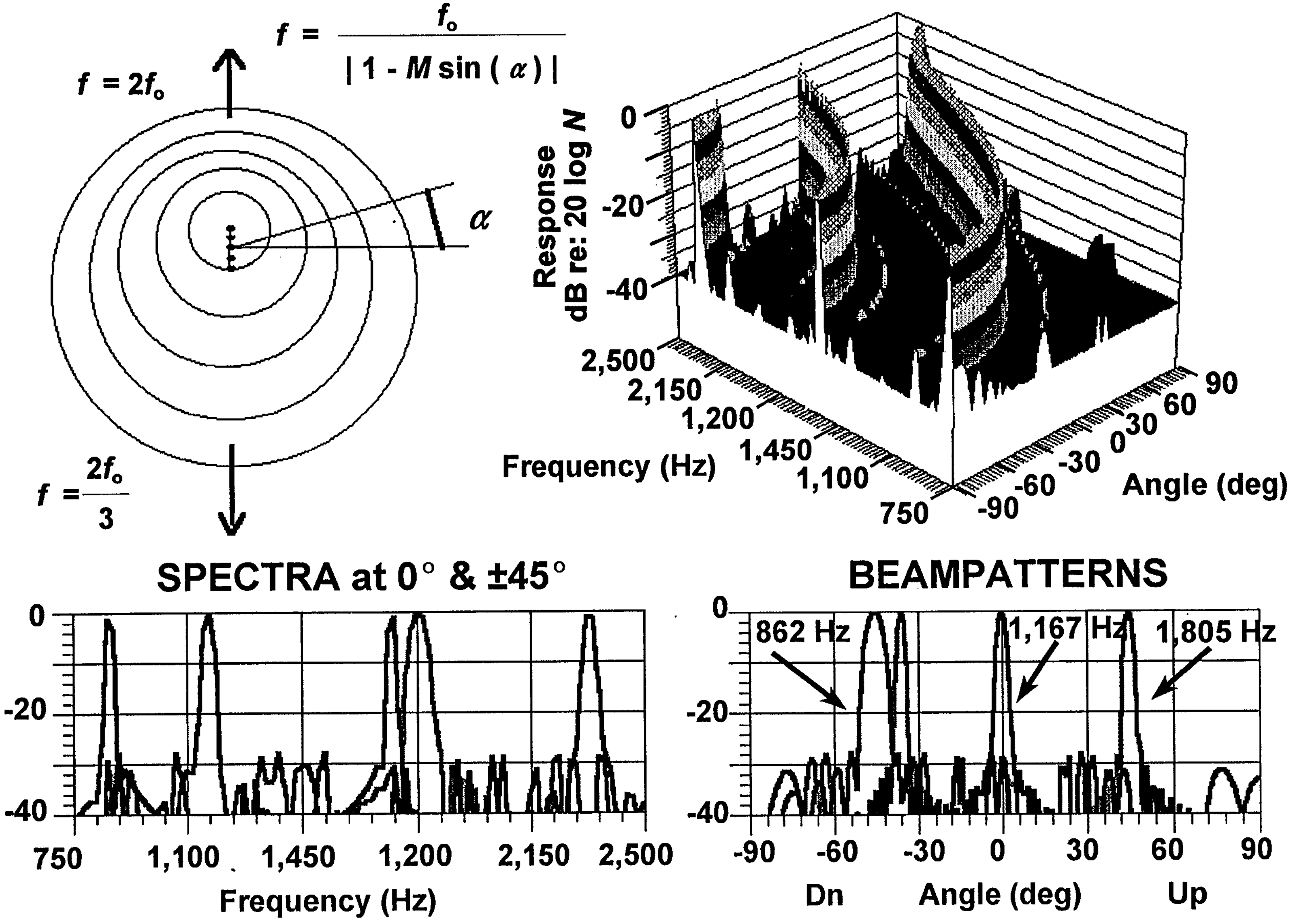

**Figure 12.** Beam-frequency spectrum, SFS mode, $M$= 0.5, $f_o$ = 1.17 kHz, shaded.

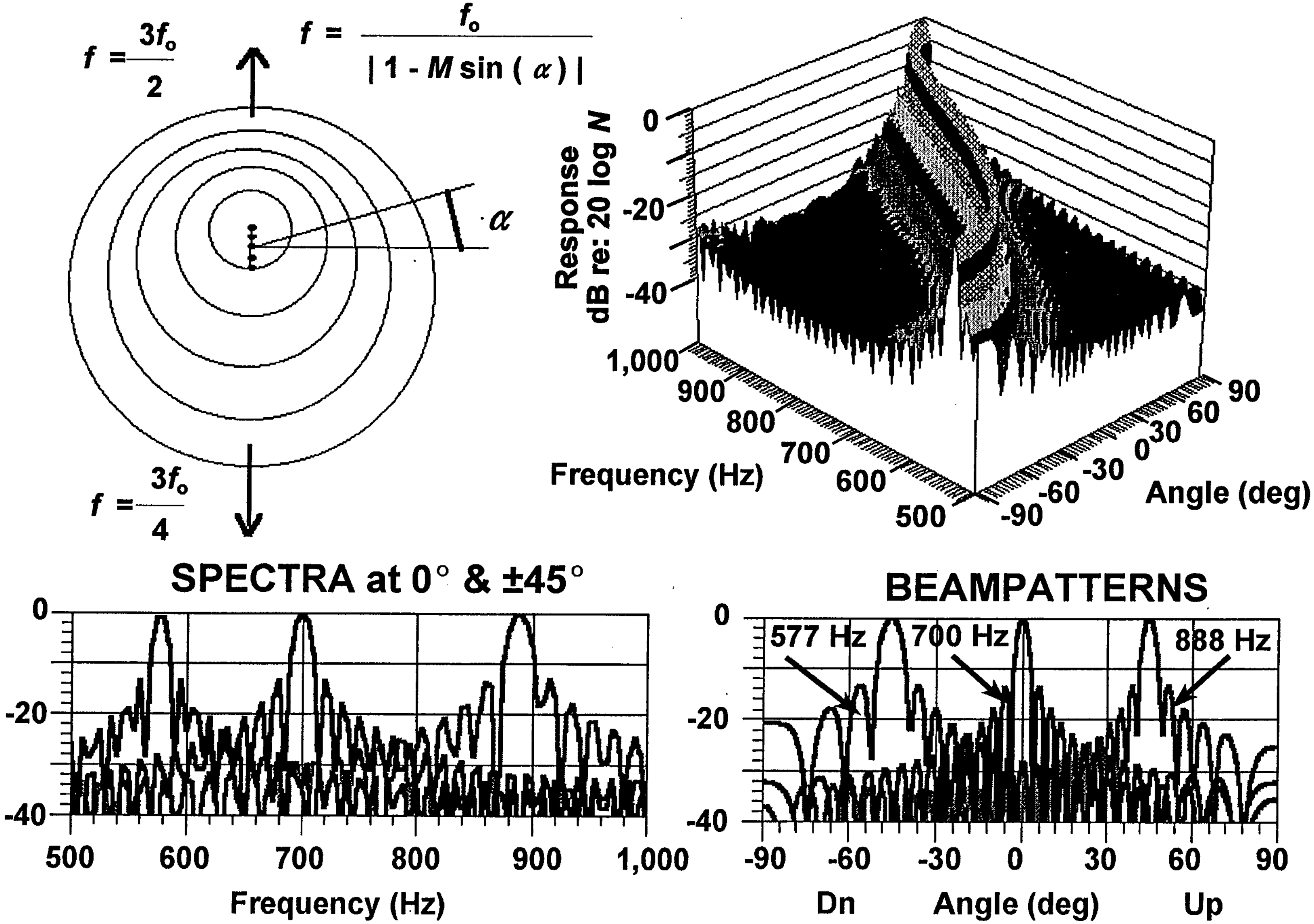

**Figure 13.** Beam-frequency spectrum, SFS mode, $M$= 0.3, $f_o$ = 700 Hz, unshaded.

expected, the beam width is larger at lower frequencies, as shown in the lower right. The band of sidelobes around –30 dB are from the discontinuities in the shading coefficients. Since the beam energy is the integral over the bandwidth, the energy source level is large.

This mode and the other broadband modes are useful in deep noise-limited situations when the target is in the beam. In shallow water or in a surface duct, the near broadside modes propagate energy to the longest range (see discussion with figure 24).

The beam-frequency spectrum for the end-fired mode fired from the bottom up at Mach 5 is shown in figure 7. Because of the slight delay, the earliest wave fronts are larger, generating a broadband narrow beam focused up at approximately $\alpha=\sin(1/M)\approx+11$ degrees. Firing from the top down will generate a similar beam focused down at approximately –11 degrees. Like the previous figure, the beam-frequency spectrum shows a broadband beam containing all frequencies. The spectrum at beam center, shown in the lower left, is flat. The beam width is larger at lower frequencies, as shown in the lower right. The beam energy source level is the same as that for the broadside mode.

The steering capability, fired top down or bottom up, provides a high-energy beam at a steering angle chosen to optimize the coverage in known oceanographic environments and tactical situations. For example, in downward-refracting conditions, a beam steered up will provide the largest area coverage, while downward beams will do better in upward-refracting environments.

The center-fired mode is treated as two end-fired arrays fired at the same time in opposite directions from the center. The $M$=5 version is illustrated in figures 8 and 9 for shaded and unshaded sources, respectively. At high frequencies, two beams are formed at $\pm 11$ degrees ($\sin^{-1}$ $(1/M)$) with peak pressure $p_{cf}(f) = \sum_{n}^{N/2} p_n(f)$. The energy source level is 6 dB below that of the end-fired modes. At low frequencies, the beams overlap, resulting in one wide broadside beam.

The center-fired mode is most useful in deep noise-limited convergence-zone environments, where the downward and upward beams ensonify targets in different parts of the convergence zone. The wider beams increase the coverage to include most ranges accessible without a bottom bounce. The array is much better than an omnidirectional signal underwater sound (SUS) because much of the SUS energy is wasted in paths that do not propagate to the target.

The unshaded modes are shown in figure 9 to illustrate that there are only slight differences between their beam patterns and sidelobe levels and those for the shaded modes. Both have higher sidelobe levels than the endfired modes. The peak sidelobes in the shaded version are about –6 dB below the unshaded version. The shaded version is recommended.

The SFS mode is obtained by firing the elements at $M < 1$, either bottom up or top down.[1] Beam-frequency spectra for a shaded bottom-up SFS mode with $M$=0.1, 0.3, and 0.5 are illustrated in figures 10–12.

The SFS mode generates multiple beams in a single ping, each at a different frequency. When $M < 1$, the wave front from an element passes the next element before its firing, as shown in the top left of each figure. This generates a set of nested circles, each centered on an element. Element displacement results in the spacing between the wave fronts varying with angle from the center of the array. For the bottom-up mode illustrated, the downward-propagating wave fronts have larger spacing. When they pass a receiver at the speed of sound, the frequency is lower. Similarly the upward-propagating wave fronts have smaller spacing, resulting in a higher frequency. The $M = 0.1$ mode, shown in figure 10, sweeps between 0.9 and 1.1 $f_o$, as illustrated in the 3-D beam-frequency spectrum in the upper left. The spectra received at 0 and $\pm 45$ degrees are shown in the lower left. The beam

[1] "Frequency Dispersive Transmitting Array," U.S. Patent No. 5,150,336, by S. F. Sullivan, F. Gordon, and B. Castile, 22 September 1992.

patterns from receivers vs angle with a 1-Hz band filter at the peak frequencies are shown in the lower left. The beams are wider than the other SFS modes because of the low frequency. Lower Mach numbers are useful for obtaining a signal in a specific frequency range; e.g., in measuring target strength as a function of frequency.

The Mach 0.3SFS (figure 11) provides unambiguous resolution in frequency and angle, with the narrowest beams for a given aperture and number of elements. The lower left illustrates the spectra received at $\alpha = 0$ and $\pm 45$ degrees. The 1-Hz band beam pattern is shown in the lower right.

The shaded $M$=0.3 SFS mode is most useful in reverberation-limited environments, as discussed with figures 26–28.

For larger Mach numbers (e.g., figure 12, $M$=0.5), the SFS mode generates multiple beams for a portion of the spectrum. These result from harmonics of the element impulses. The ambiguities lead to overlap of the target and reverberation returns, reducing the echo-to-reverberation ratio from the $M = 0.3$ SFS mode at many ranges. In this $M = 0.5$ example, frequencies above 1,625 Hz (for angles > 0 degrees) have ambiguities. SFS modes with $0.3 < M < 1$ are not recommended except in very special environments.

Figure 13 shows the beam-frequency spectrum for an unshaded array at Mach 0.3. The beam patterns and spectra have the sin(x)/x shape typical of uniform shading, with the first sidelobe at -13 dB. These high sidelobes reduce the shallow-water echo-to-reverberation ratio from that obtained with the shaded version of figure 11.

## BANDWIDTHS AND RESOLUTIONS

For $M < 1$, the frequency components of $p_n(t)$ add in phase at angle $\alpha$, forming a beam at frequency $f(\alpha)$. This mode is illustrated in figure 11 for $M = 0.3$. It is called the steered-frequency source (SFS) mode because different beams have different frequency components given by Eq. 13. At each angle, $N$ pulses spaced at $t$=1/$f$ are received. The bandwidth of the received signal is the inverse of the signal duration:

$$\Delta f_{SFS} = \frac{1}{T} = \frac{f(\alpha)}{N-1} \qquad (14)$$

giving a signal with time-bandwidth product equal to 1.[2] The beamwidth is given by

$$\Delta \alpha_{SFS} = \frac{\lambda}{D \cdot \alpha \cos} = \frac{c}{f(\alpha) \cdot (N-1) \cdot d \cdot \alpha \cos} \qquad (15)$$

For $M \geq 1, f(\alpha)$ is undefined at

$$\alpha_M = \sin^{-1}\left(\frac{1}{M}\right), \quad M \geq 1 \qquad (16)$$

As illustrated in figure 7 for $M = 5$, all the signals $p_n(t)$ superimpose at $\theta_M$, giving a single beam with the full bandwidth of $p_n(t)$. The beam angle is varied by changing the mach number, which is done by changing the interelement time delay, $t_n$. The beamwidth varies with frequency according to

$$\Delta \alpha_M = \frac{\lambda}{D \cdot \cos(\alpha_M)} = \frac{c}{f \cdot (N-1) \cdot d \cdot \cos(\alpha_M)} \qquad (17)$$

[2] Signals with $TW > 1$ can be achieved by varying the element spacing and changing the array timing from that of a constant Mach number. In this case, each beam has a different time compression.

# SENSITIVITY TO TIMING, SPACING, AND SHADING ERRORS

This section analyzes the sensitivity of the beam-frequency spectra sidelobe levels to errors in timing, spacing, and shading coefficient.

Assume that the realized values of the element positions, firing times, and shading coefficients are given by

$$d_n = n\hat{d}(1 + \frac{\delta_{dn}}{n}) \tag{18}$$

$$t_n = \frac{n}{M}\left\{\frac{\hat{d}}{c}(1 + \frac{M\delta_{tn}}{n})\right\} \tag{19}$$

and

$$\alpha_n = \hat{\alpha}_n (1 + \delta_{\alpha n}) \tag{20}$$

where $\hat{d}$ and $\hat{\alpha}_n$ are the design values of the spacing and shading coefficients, $\delta_{dn}$ is the error in position of the $n^{\text{th}}$ element expressed as a fraction of the design spacing, $\delta_{tn}$ is the error in the time of the $n^{\text{th}}$ element expressed as a fraction of the design travel time between the elements, and $\delta_{\alpha n}$ is the error in the shading coefficient of the $n^{\text{th}}$ element expressed as a fraction of the design coefficient.

The beam-frequency spectrum from Eq. 11 with errors becomes

$$B(f, \theta) = \frac{\left|\sum_{n}^{N} \hat{\alpha}_n (1 + \delta_{\alpha})e^{j2\pi f\frac{n\hat{d}}{Mc}\left((1 + \frac{M\delta_{tn}}{n}) - M(1 - \frac{\delta_{dn}}{n}) \sin\theta\right)}\right|}{N^2} \tag{21}$$

The beam-frequency spectra shown below are calculated from Eq. 21 using a random-number generator with a uniform distribution.

Figures 14 and 15 show beam frequency spectra for 10% errors in shading coefficient, timing, and spacing for the broadside mode.

Figure 14 is obtained using the random-number generator to calculate errors which remain unchanged throughout the calculation. This is the beam-frequency spectrum which would be obtained from one realization of the errors.

Figure 15 is generated by having a new set of errors generated for every angle and frequency point in the plot. It is useful in illustrating the variability that might be obtained from realization to realization. The spread in sidelobe levels in figure 15 is representative of the spread that would be obtained from several realizations. Note that peak sidelobe levels around –25 dB are obtained with 10% errors. Figures 14 and 15 should be compared with figure 6, which shows the shaded broadside mode with no errors.

Figures 16 and 17 illustrate the beam-frequency spectra achieved with one and multiple realizations of 10% errors for the SFS mode with $M = 0.3$.

Figure 16 is obtained using the random-number generator to calculate errors which remain unchanged throughout the calculation. This is the beam-frequency spectrum which would be obtained from one realization of the errors.

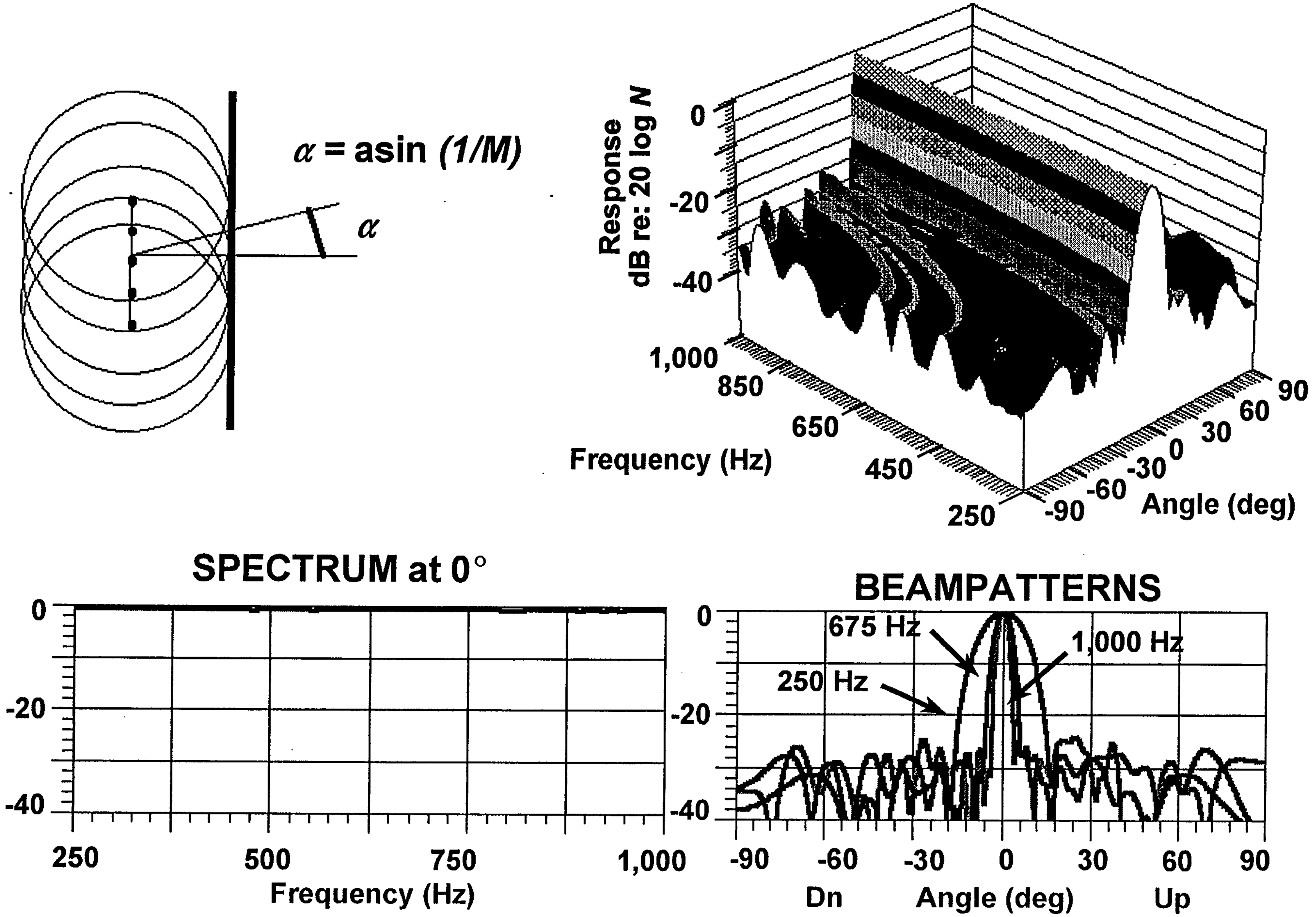

**Figure 14.** Beam-frequency spectrum, broadside mode, $M = \infty$ , shaded, 10% weight, timing and spacing errors, one realization.

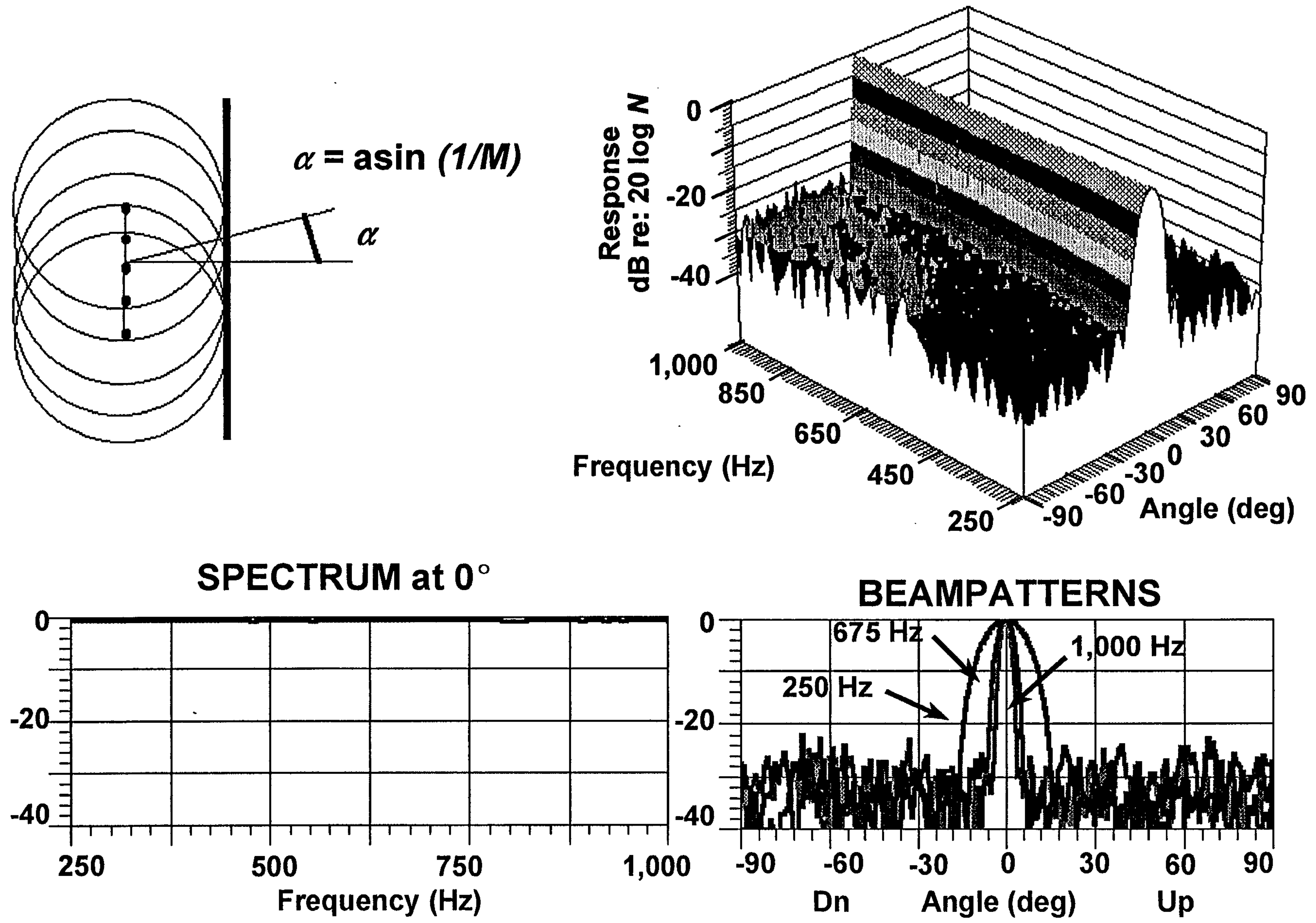

**Figure 15.** Beam-frequency spectrum, broadside mode, $M = \infty$, shaded, 10% weight, timing and spacing errors, multiple realizations.

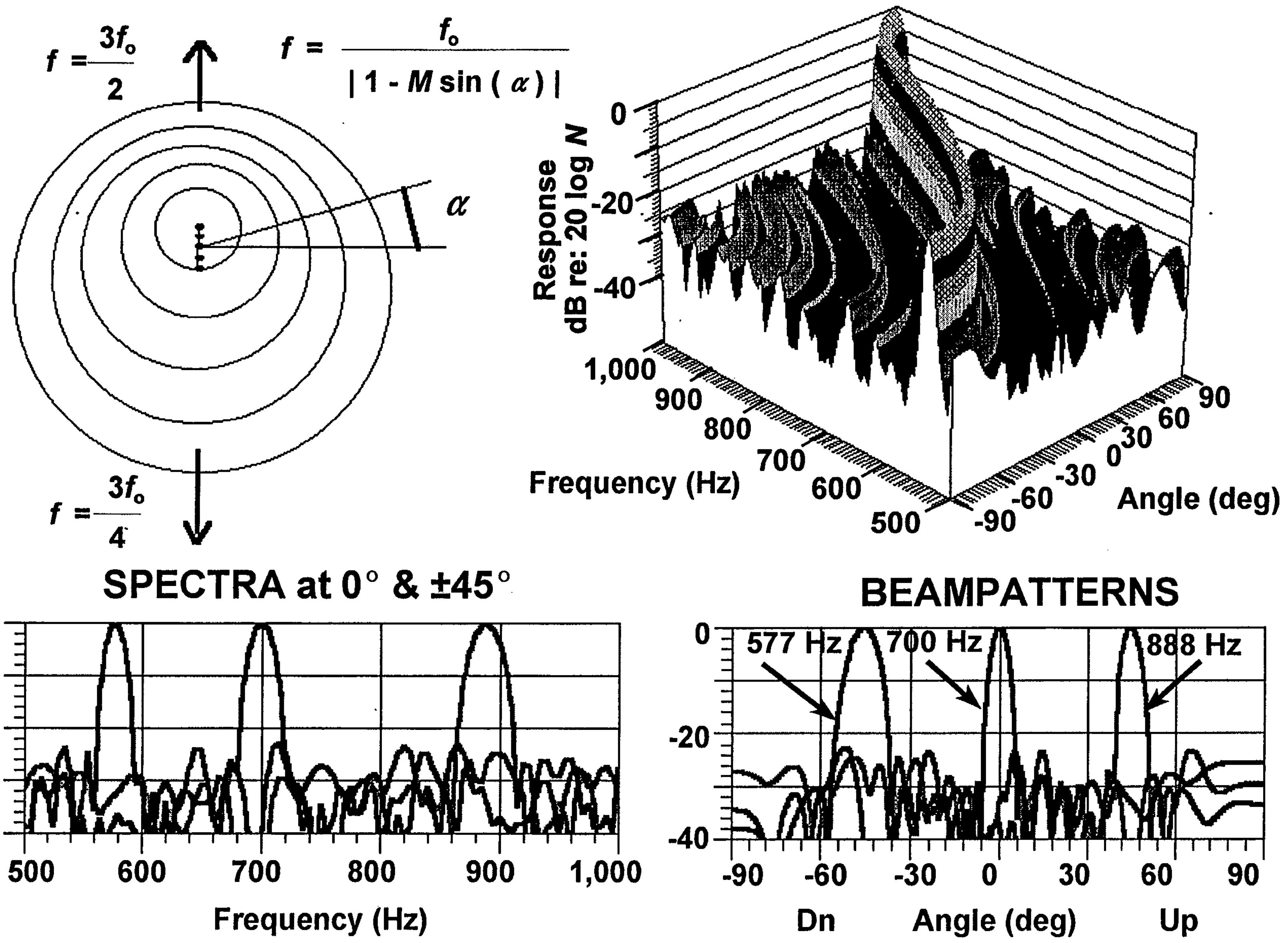

**Figure 16.** Beam-frequency spectrum, SFS mode, $M = 0.3$, $f_o = 700$ Hz, shaded, 10% weight, timing and spacing errors, one realization.

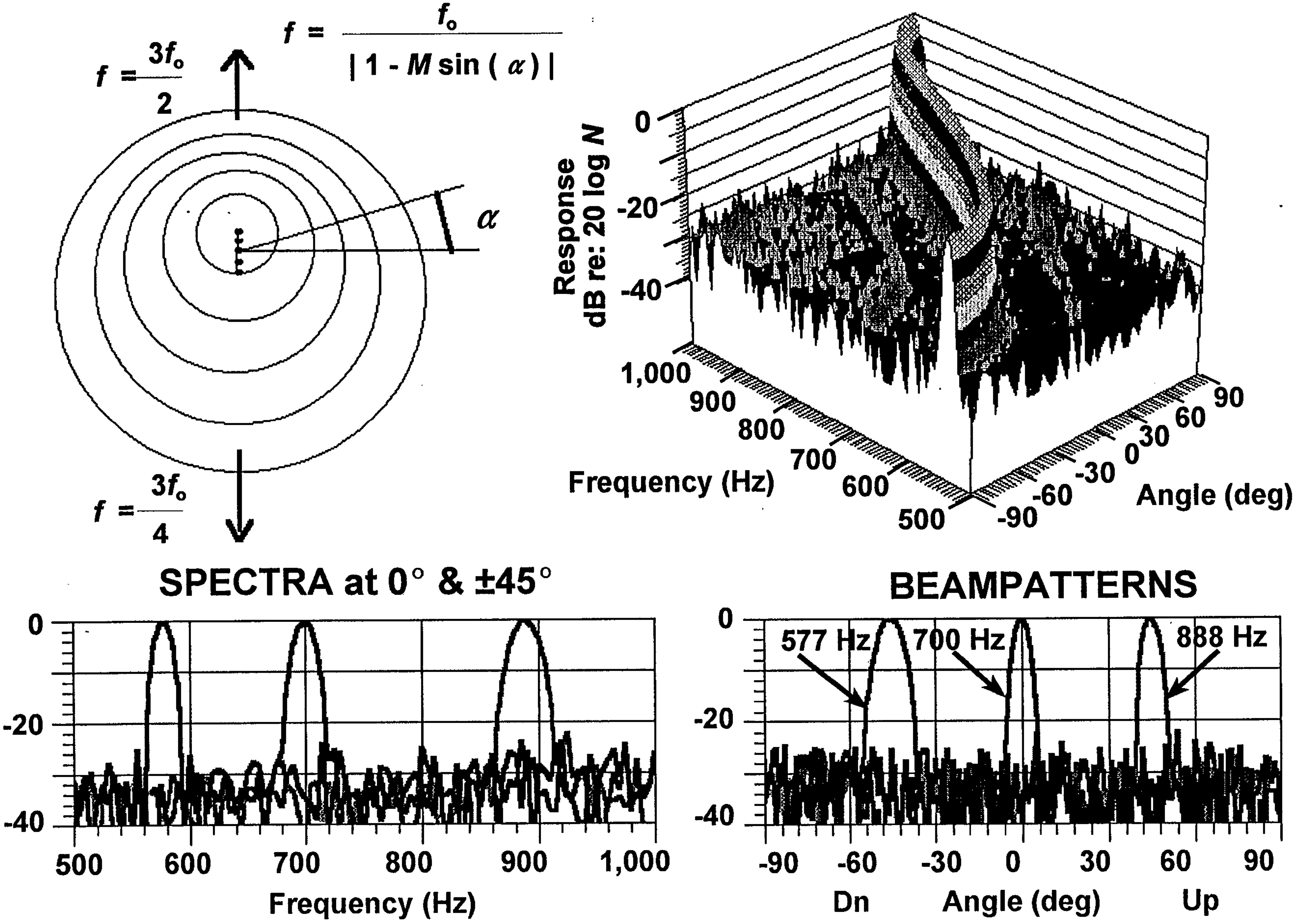

**Figure 17.** Beam-frequency spectrum, SFS mode, $M = 0.3$, $f_o = 700$ Hz, shaded, 10% weight, timing and spacing errors, multiple realizations.

Figure 17 is generated by having a new set of errors generated for every angle and frequency point in the plot. The spread in sidelobe levels in figure 17 is representative of the spread that would be obtained from several realizations. As in the broadside mode, peak sidelobe levels around –25 dB are obtained with 10% errors. Figures 16 and 17 should be compared with figure 11, which shows the shaded SFS mode with $M = 0.3$ with no errors.

# SELECTION OF ELEMENT WEIGHTS TO ACHIEVE SHADING COEFFICIENTS

This section describes the method of choosing the weights of each element in the multimode source array to achieve the desired pressure shading coefficients.

An empirical method for estimating source levels achieved from TNT charges is summarized in Urick.[3] A more thorough review of the literature has been summarized by Gottwald.[4] Urick's equations will be used because they are more available and differ from Gottwald's by only a few dB in the 500–1,000-Hz band. We will consider only the shock wave portion of the energy, because the energy and times of the bubble components are depth dependent and are likely to be influenced by the presence of the other charges. For shallow sources above a few hundred hertz, all the source elements should have fired well before the first bubble has formed.

Energy source level estimates will be lower than if the bubble component is used. However, since the bubble component timing is influenced by water depth and the presence of the other charges, its contribution is not likely to match the model. The bubble effects will affect the sidelobe levels at low frequencies.

The shock wave portion of the pressure pulse from an explosive charge of weight $w_n$ pounds is given by (see Appendix A)

$$p_n(t) = p_{on} e^{-t/t_{0n}} \tag{22}$$

where

$$\begin{aligned} p_{on} &= \gamma \cdot w_n^{0.377} \ \mu \text{ Pa at 1 m} \\ \gamma &= 2.14 \times 10^{13} \end{aligned} \tag{23}$$

and

$$\begin{aligned} t_{on} &= \beta \cdot w_n^{0.26} \text{s} \\ \beta &= 2.07 \times 10^{-4} \end{aligned} \tag{24}$$

The Fourier transform of the pressure is given by

$$p_n(f) = \frac{\sqrt{2}\, p_{on}}{1/t_{on} + j2\pi f} \tag{25}$$

The energy-density spectrum

$$|p_n(f)|^2 = \frac{2p_{on}^2}{(1/t_{on}^2 + 4\pi^2 f^2)} \quad \mu\text{Pa}^2 s^2 \text{ at 1 m} \tag{26}$$

is plotted in figure 18.

[3] Urick, R. J., *Principles of Underwater Sound*, 3rd edition, McGraw–Hill, 1983, pp. 86–96.
[4] Gottwald, J. T., "Technical Note: SUS Sound Source Level Considerations," G/J Associates, 18 May, 1992.

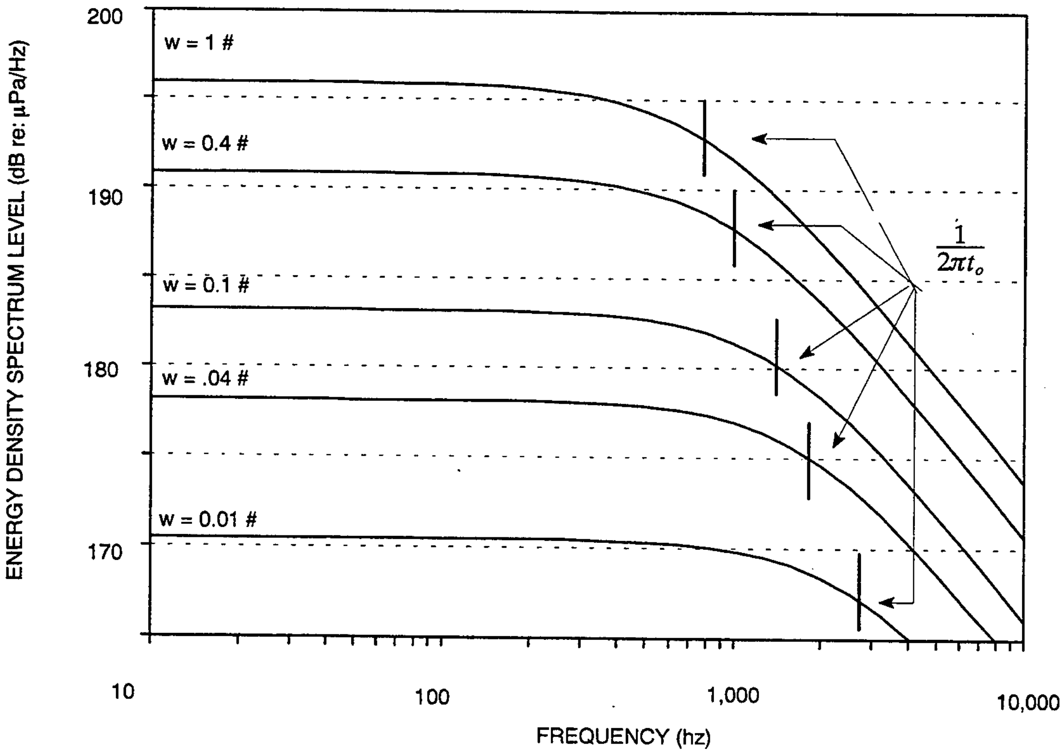

**Figure 18.** Energy-density spectrum of point charges of TNT for various weights.

The approximation that all $p_n$ $(f)$ are identical except for an amplitude shading coefficient is valid for frequencies less than the transition frequency, defined by

$$f < f_{tn} = \frac{1}{2\pi t_{on}} \tag{27}$$

The points where $f = f_{tn}$ are indicated with vertical lines in the figure. At higher frequencies, the source levels will drop and the phase will vary from element to element with the weight causing degradation. We will proceed assuming Eq. 27 is valid and then examine the beam patterns in the vicinity of $f = f_{tn}$ to estimate the effects of this assumption.

The shading coefficients $\alpha_n$ are chosen as a tradeoff between beam width, sidelobe level, and ease of implementation, as described in the previous section. The problem addressed in this section is to determine the element charge weights, $w_n$, given that the source design is constrained to a total weight $w_T$:

$$\sum_{n=1}^{N} w_n = w_T \tag{28}$$

Using Eq. 23, 24, and 25 for $f < < \frac{1}{2\pi t_{on}}$, we obtain

$$p_n(f) = \sqrt{2}\,\gamma\beta w_n^{0.637} = a_n\, p_e(f) \tag{29}$$

The last step uses Eq. 8. Summing both sides over *n* and using Eq. 9 and 28, it can be shown that

$$p_e(f) = \frac{\sqrt{2}\ \gamma\ \beta\ w_T^{0.637}}{\left(\sum_{n=1}^{N}(\alpha_n^{1.57})\right)^{0.637}} \approx \sqrt{2}\ \gamma\ \beta\ \left(\frac{w_T}{N}\right)^{0.637} \tag{30}$$

and

$$w_n = \frac{w_T\, \alpha_n^{1.57}}{\sum_{n=1}^{N} \alpha_n^{1.57}} \tag{31}$$

# BEAM-FREQUENCY ENERGY-DENSITY SPECTRA

The beam-frequency spectra described above assumed that the energy-density spectra of the elements were identical, with the exception of an amplitude shading coefficient. This section examines the exact multimode source characteristics which are achieved using source elements of various weights in the vicinity of the transition frequency, $f_{tn}$, where both the amplitude and phase of the elements vary with weight.

Beam-frequency energy-density spectra (without normalization) near the transition frequency are calculated using Eq. 6 and 7 and the complex Fourier transform of the pressure, Eq. 25:

$$p^2(f) = \left| \sum_{n=1}^{N} \frac{\sqrt{2}\, p_{on}}{1/t_{on} + j 2\pi f} \; e^{-j 2\pi f \frac{nd}{Mc}(1-M \sin \theta)} \right|^2 \qquad (32)$$

SFS and broadside modes without errors in weight, timing, and spacing are shown in figures 19 and 20. The total weight is chosen to be $w_T = 52.8$ pounds in order to achieve $f_{tn}(a_n = 1) = 750$ Hz, putting the transition frequency for unity shading coefficient in the middle of the 500–1,000 Hz. band.

Note that the beam characteristics and sidelobe levels are maintained well past $f_{tn}$, as in the approximate case shown in figures 6 and 11. Also note the slow decrease in energy density with increasing frequency, illustrated best in the spectrum at 0 degrees in figure 20. This characteristic is much different from that of the individual elements shown in figure 18, but is consistent with the spread in element weights and the associated spread in $f_{tn}$ from 642 to 1,170 Hz.

The method of calculating the weights from the shading coefficients, Eq. 31, appears to give the desired –30-dB sidelobe characteristics throughout an octave band centered on $f_{tn}$. The approximations of the previous beam-frequency spectra calculations are valid for frequencies below 1.3 $f_{tn}$.

Figures 21 and 22 show the same beam-frequency energy-density spectra as figures 19 and 20, with multiple realizations of 10% errors in element weight, spacing, and timing. Note that the sidelobe peaks increase and that the main lobe decreases with increasing frequency. At the highest frequency, they remain in the vicinity of –25 dB, as in the approximate analysis.

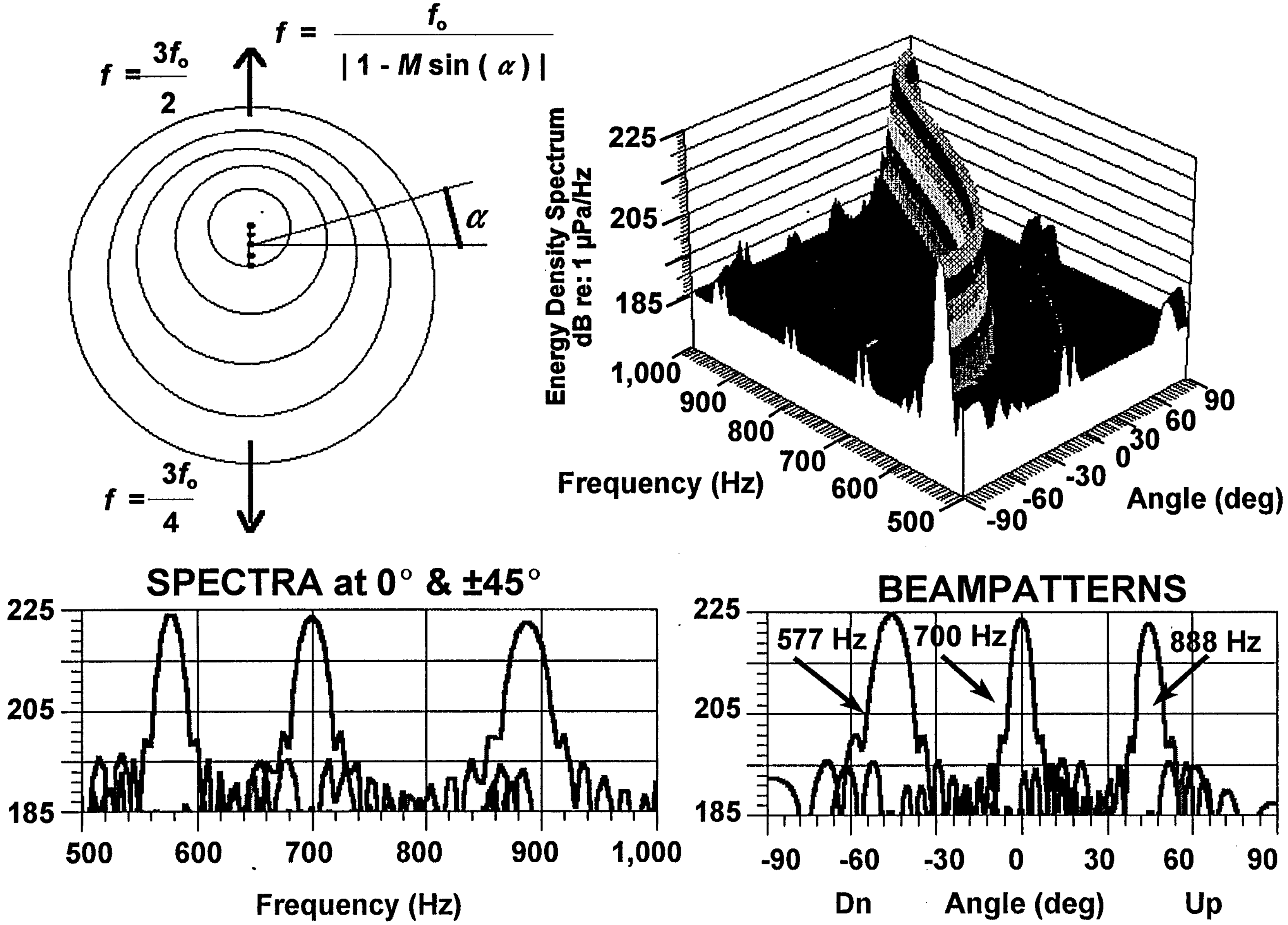

**Figure 19.** Beam-frequency energy-density spectrum, SFS mode, $M$ = 0.3, $f_o$ = 700 Hz, shaded, $W_T$ = 52.8 pounds, $f_o$ = 750 Hz, no errors.

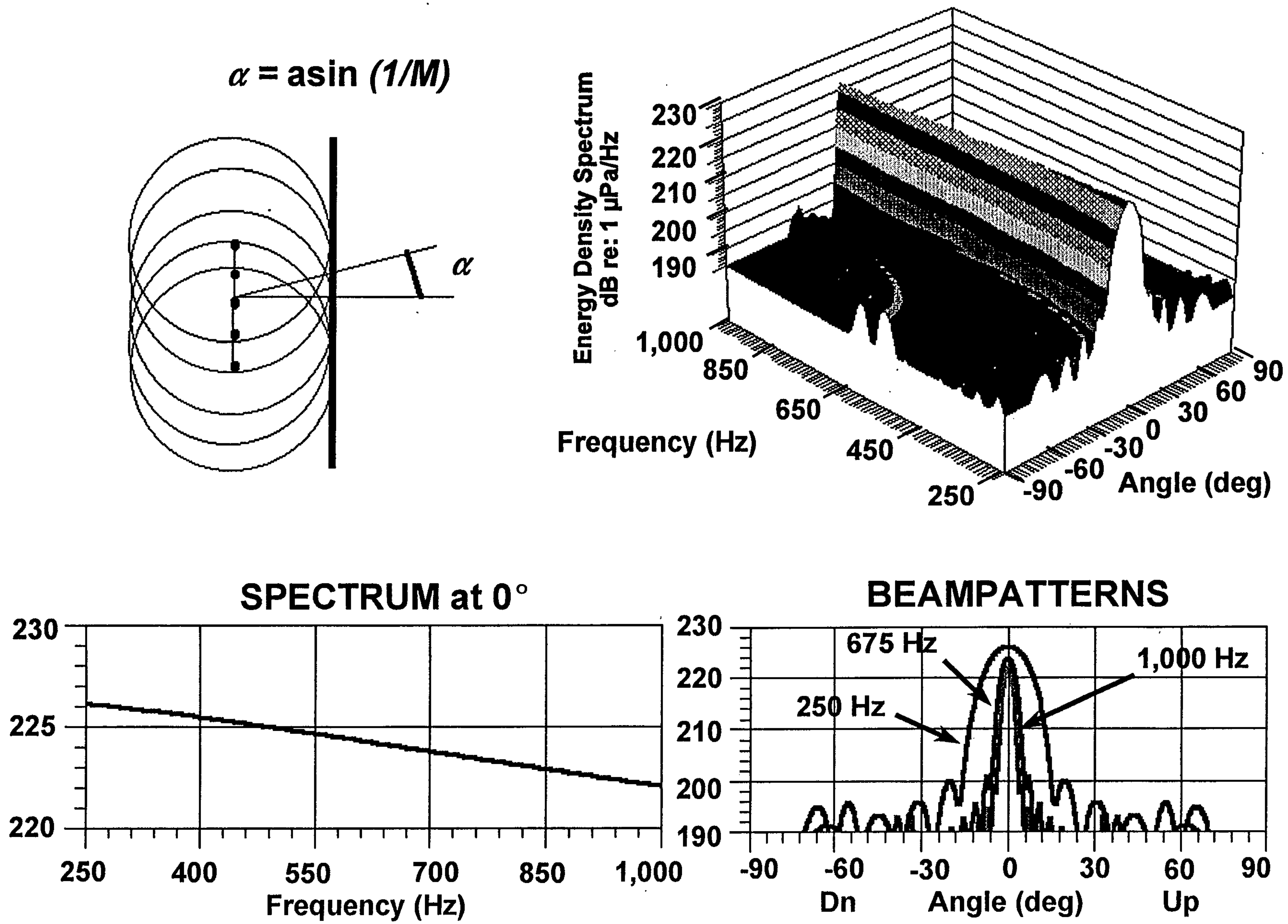

**Figure 20.** Beam-frequency energy-density spectrum, broadside mode, $M = \infty$, shaded, $W_T = 52.8$ pounds, $f_o = 750$ Hz, no errors.

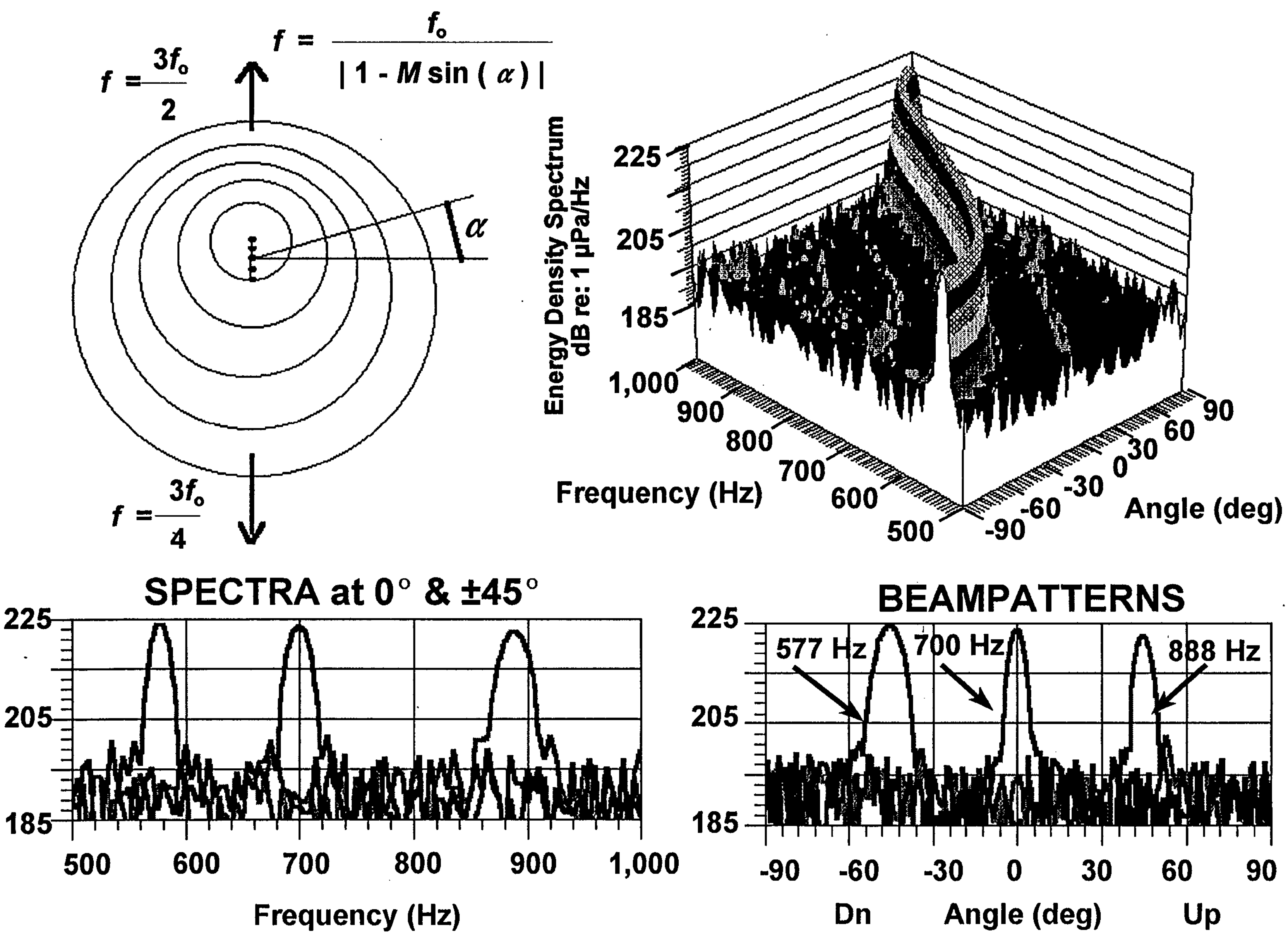

**Figure 21.** Beam-frequency energy-density spectrum, SFS mode, $M = 0.3$, $f_o = 700$ Hz, shaded, $W_T = 52.8$ pounds, $f_o = 750$ Hz, 10% errors, multiple realizations.

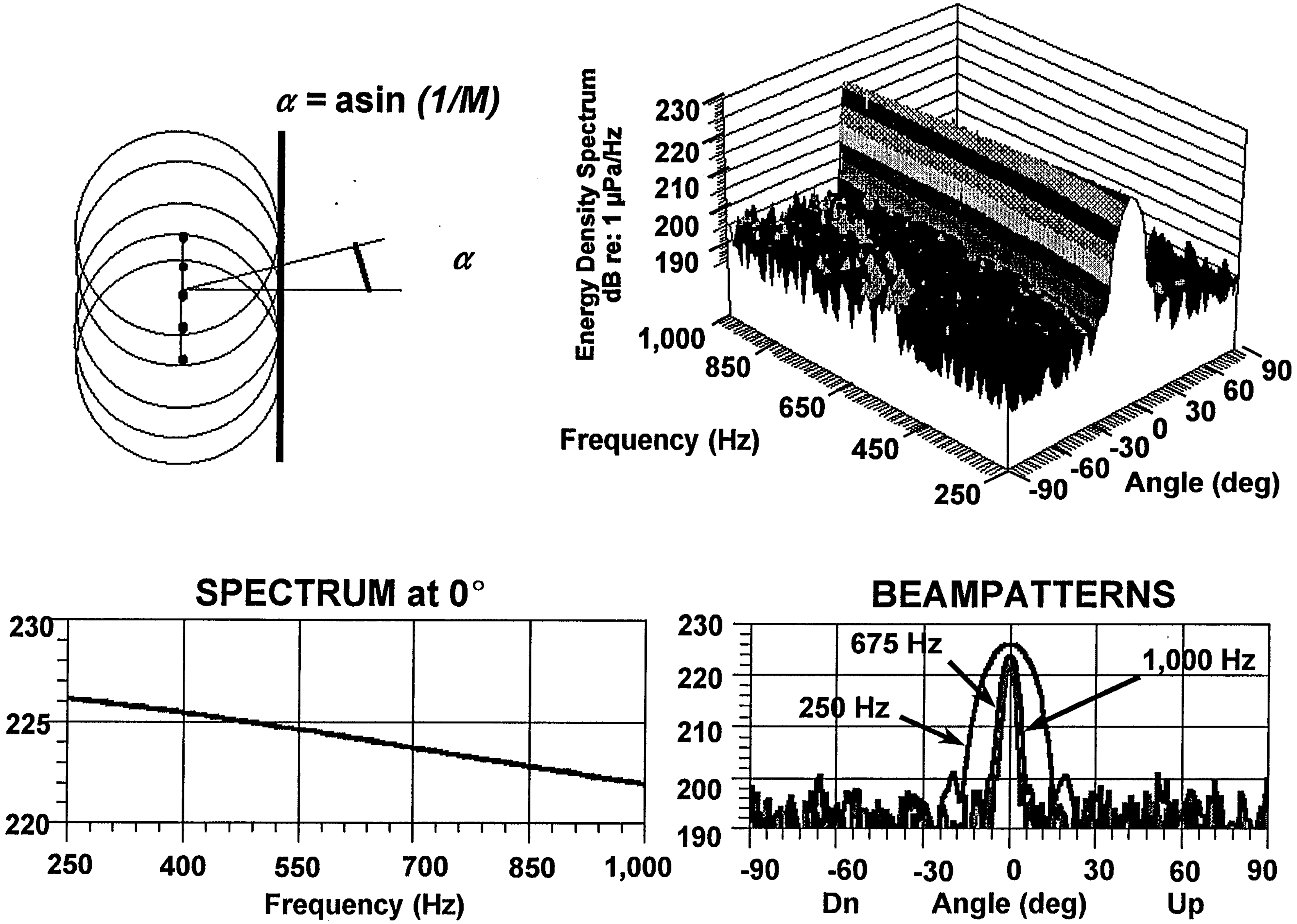

**Figure 22.** Beam-frequency energy-density spectrum, broadside mode, $M = \infty$, shaded, $W_T$ = 52.8 pounds, $f_o$ = 750 Hz, 10% errors, multiple realizations.

# ESTIMATE OF SOURCE LEVEL

The total energy on a beam, $E_S$, in units of µPa$^2$ s, is given by

$$E_s = \int_0^T p_b^2(t)dt = \int_{fc-\frac{W_S}{2}}^{fc+\frac{W_S}{2}} p_b^2(f)\,df \qquad \mu\mathrm{Pa}^2\mathrm{s} \tag{33}$$

where $W_S$ is the bandwidth of the source used in the receiver and $f_o$ is the center frequency.[5] For the broadband source modes ($M > 1$), $W_S$ is the receiver bandwidth, $W_r$. For SFS modes ($M < 1$), $W_S$ is chosen to be the bandwidth of a beam, $\Delta f_{SFS}$, given by Eq. 14.

As is seen in figures 19–22, the spectra are slowly varying, $p^2(f) \approx p^2(f_c)$ over $W_S$. The beam energy is approximated by

$$E_s = p_b^2(f_c)\,W_s \qquad \mu\mathrm{Pa}^2\mathrm{s} \tag{34}$$

$f_C$ is the center frequency of the receiver band. On beam axis, using Eq. 8, 9, and 10

$$E = N^2 p_e^2(f_c)\,W_s \qquad \mu\mathrm{Pa}^2\,\mathrm{s} \tag{35}$$

The weighted mean element energy-density spectrum, $p_e{}^2(f_C)$, can be obtained from Eq. 30 for a given total weight, $w_T$.

For the end-fired and broadside modes, all frequencies received contribute to the energy in the main beam, resulting in an energy source level given by

$$ESL_{EF} = 10 \log p_e^2(f_c) + 10 \log W_r + 20 \log N \qquad dB\ re : \mu\mathrm{Pa}^2\mathrm{s} \tag{36}$$

For the center-fired mode, only $N/2$ sources contribute to any beam, and the energy source level on beam axis is given by

$$ESL_{CF} = 10 \log p_e^2(f_c) + 10 \log W_r + 20 \log \frac{N}{2} = ESL_{EF}-6 \qquad dB\ re : \mu\mathrm{Pa}^2\mathrm{s} \tag{37}$$

For the SFS mode, the bandwidth of the signal on any given beam is given by Eq. 14 and the ESL at angle $\alpha$ becomes

$$ESL_{SFS}(\alpha) = 10 \log p_e^2(f(\alpha)) + 10 \log f(\alpha) + 10 \log N \qquad dB\ re : \mu\mathrm{Pa}^2\mathrm{s} \tag{38}$$

Figure 23 shows the estimate of the energy source levels that would be achieved with the array illustrated in figures 6–17 as a function of the total weight of TNT in the octave band centered at 750 Hz. The transition frequency is also shown vs total weight for a 48-element array. An octave receiver bandwidth is assumed for the $M > 1$ modes, $W_r = f_C$.

[5] We are using units that assume $\varrho c = 1$ where $\varrho$ is the density of water and $c$ is the speed of sound. This works out when source levels are expressed in dB re: µPa. The pressure squared, $p^2(t)$, has units of µPa$^2$ and the energy-density spectrum, $p^2(f)$, has units of µPa$^2$/Hz$^2$. Units of intensity (energy per unit area at range of 1 meter) can be obtained by dividing Eq. 33 by $\varrho c$.

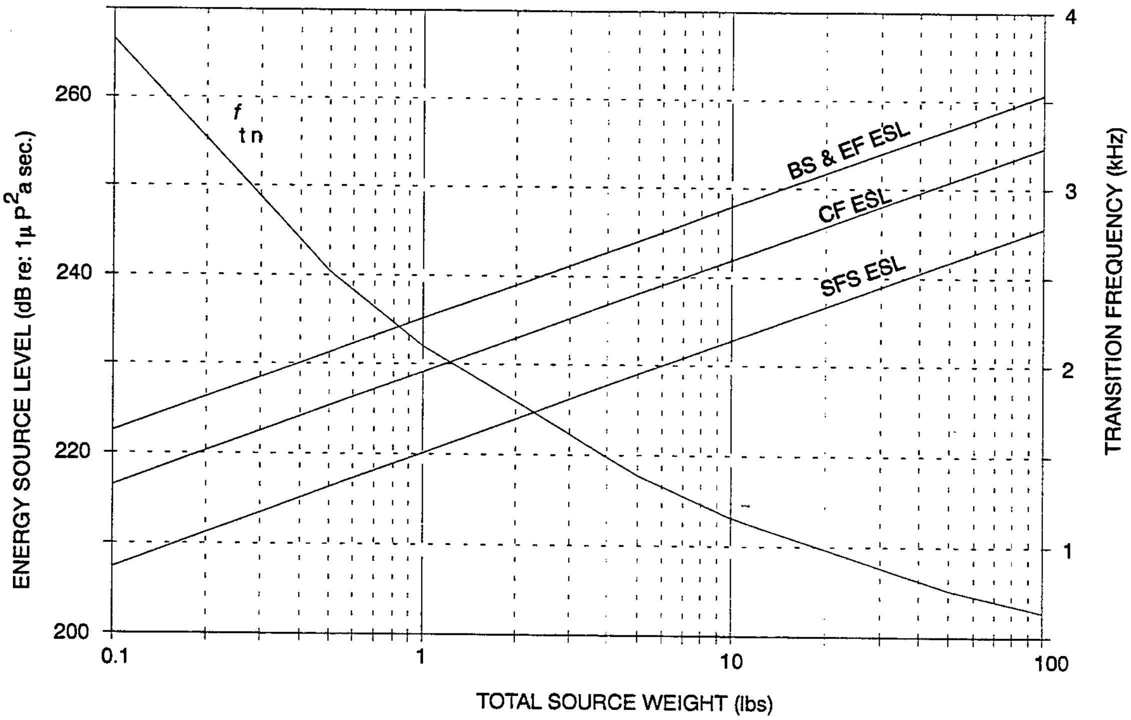

**Figure 23**. Estimated energy source levels vs total weight of TNT for $N = 48$ for the labeled modes. The transition frequency $f_{tn}$ is also shown.

$ESL_{SFS}$ is smaller than the end-fired and broadside modes by approximately $10 \log N$. This can be interpreted two ways:

1. Considering the beam-frequency spectrum, the bandwidth of the SFS mode is limited, while modes with $M > 1$ are limited only by the bandwidth of the receiver. For the broadband modes, all the energy is being focused at a given angle.

2. Like an omnidirectional source, the SFS mode spreads energy in all angles; thus the total energy is not focused. Unlike an omnidirectional source which spreads energy at all angles and frequencies, the SFS mode distributes the energy in frequency and angle according to Eq. 13.

## APPLICATION EXAMPLES

This section presents a few illustrations of the benefits of being able to chose the mode and depth of the multimode source. Multiple source–receiver fields should be analyzed to determine which source and receiver geometries provide the best surveillance as a function of the tactical and environmental scenario.

### BROADSIDE MODE (*M* = ∞)

The broadside mode (see figure 6 for the beam-frequency spectrum) is particularly useful in maximizing energy on targets at long range in a surface duct environment, as illustrated in figure 24. Thirty rays are traced within the ±2-degree beamwidth at 0-degree elevation for a source deployed in the middle of the duct at a depth of 250 feet. The energy is focused in the duct, with leakage at the lower frequencies, where the beam is broader. Directive receivers could also be placed in the duct. Receivers near the target would minimize the target to receiver propagation loss and control surface reverberation.

The broadside mode is also useful in shallow-water environments, where the broadside beam propagates to the target with least loss (fewest bottom bounces).

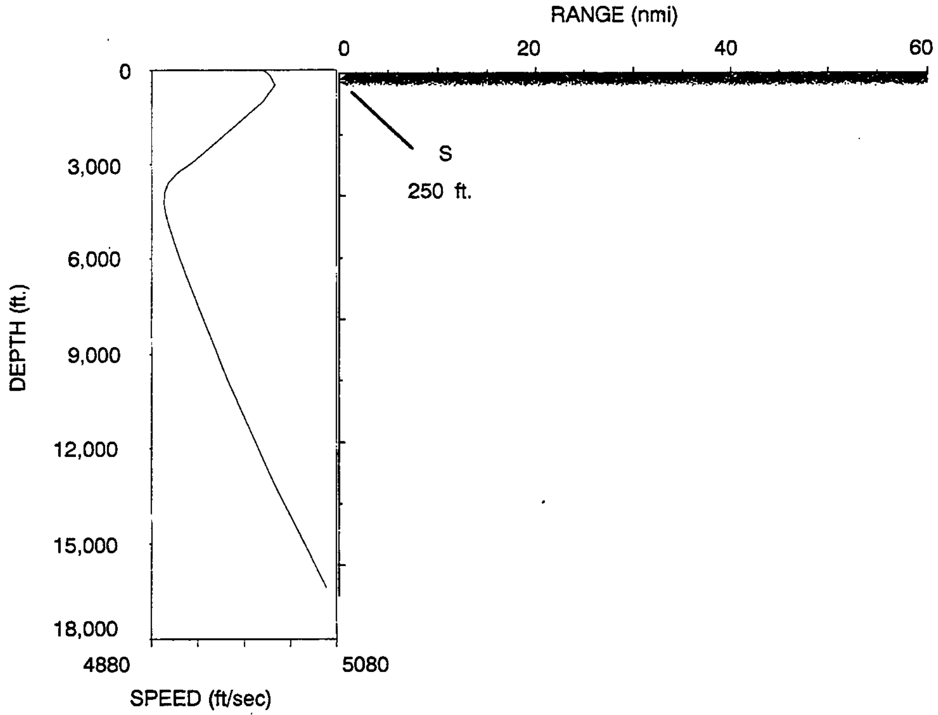

**Figure 24.** Ray trace for broadside mode (*M* = ∞) in surface duct. The 4-degree beamwidth focuses the energy in the surface duct.

### CENTER-FIRED MODE (*M* = 5.8)

The center-fired mode (see figure 8 for the beam-frequency spectrum) is particularly useful in maximizing energy on targets at long range in deep convergence-zone environments, an example of which is illustrated in figure 25. This environment is the same as in figure 24. In this case, the objective is to search for targets below the surface duct. The source is programmed to fire at a depth of 3,000 feet in the center-fired mode at $M$ = 5.8, forming two ± 4-degree beams centered at ± 10 degrees. Thirty rays are traced in each of the beams, illustrating that the source ensonifies targets below the duct at different ranges with a combination of upward and downward beams. The two beams double the coverage at the expense of 6 dB in energy source level. Directive receivers near the source in a quasimonostatic configuration would receive echoes on the similar paths.

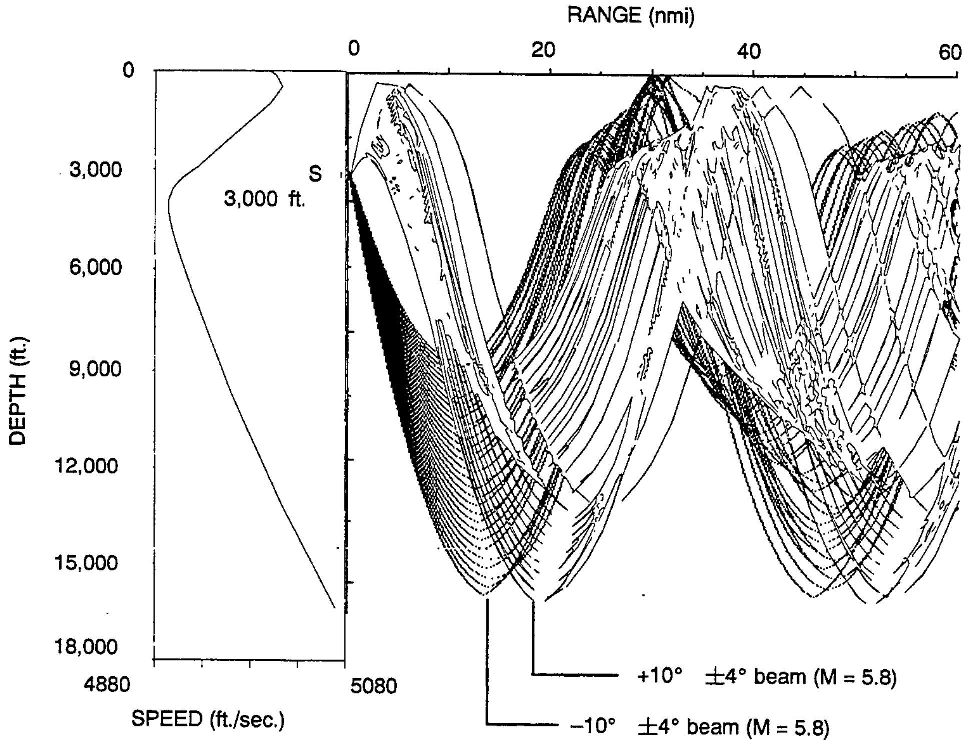

**Figure 25**. Ray trace for center-fired mode ($M$ = 5.8) forming two beams at ± 10 degrees deployed at 3,000 feet to search for targets below surface duct.

## STEERED FREQUENCY SOURCE MODE ($M$ = 0.3)

The steered frequency mode with $M$ = 0.3 forms multiple beams, each at a different frequency, as shown in the SFS beam-frequency spectrum in figure 11. This SFS mode is useful in reducing reverberation in shallow-water areas with downward-refracting sound-speed profiles.

Figure 26 illustrates 20 paths within the –5-degree beam obtained by the array at 675 Hz. The beam pulse has a bandwidth of 14 Hz and a range resolution of approximately 300 feet, as illustrated by the vertical line in the figure. When the beam pulse is at target depths, no main beam rays are interacting with the bottom. Bottom reverberation in the vicinity of the target is then limited to energy in the sidelobes of the beam. Directional monostatic and bistatic receivers combined with the range resolution of the pulse provide echo-to-reverberation-ratio improvement over an omnidirectional pulse by as much as 20 dB.

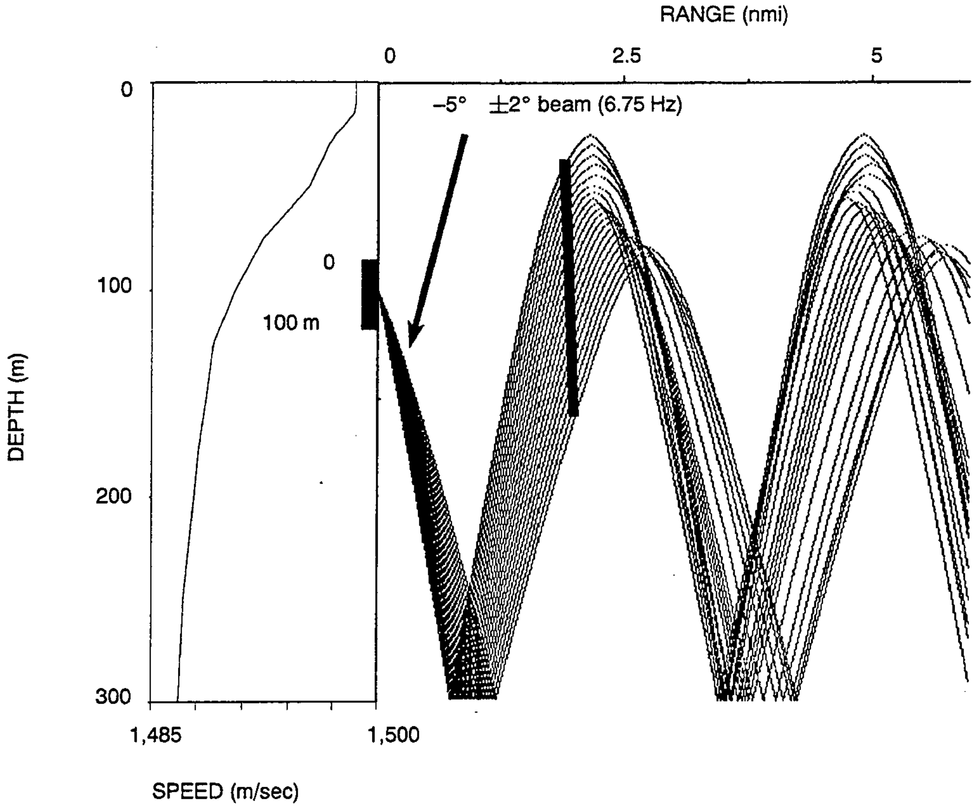

**Figure 26**. Ray trace for SFS mode (*M* = 0.3) 675-Hz beam at –6 degrees. The source is deployed at a 100-meter depth in water that is 300 meters deep. The sound-speed profile is downward-refracting.

The processing consists of frequency analysis of the received signals, producing a spectrum with a 14-Hz bandwidth every 70 ms. The predicted output from a reverberation model is plotted vs time, as shown in figure 27 for an omnidirectional monostatic receiver under the conditions shown in figure 26. Bottom reverberation from paths leaving the source downward is seen between 528 and 700 Hz and from upward paths between 700 and 1,000 Hz. Multiple bottom bounces arrive later in time. The bottom bounce from the beam skimming the surface arrives just after 5 seconds.

Two targets are modeled, each at a 100-meter depth and with a 50-dB target strength, at ranges of 1 and 2.5 nmi. The echoes arrive at 2.2 and 6.2 seconds, respectively. Note that the peaks in the echo occur at minima of the reverberation resulting from the propagation described in figure 26.

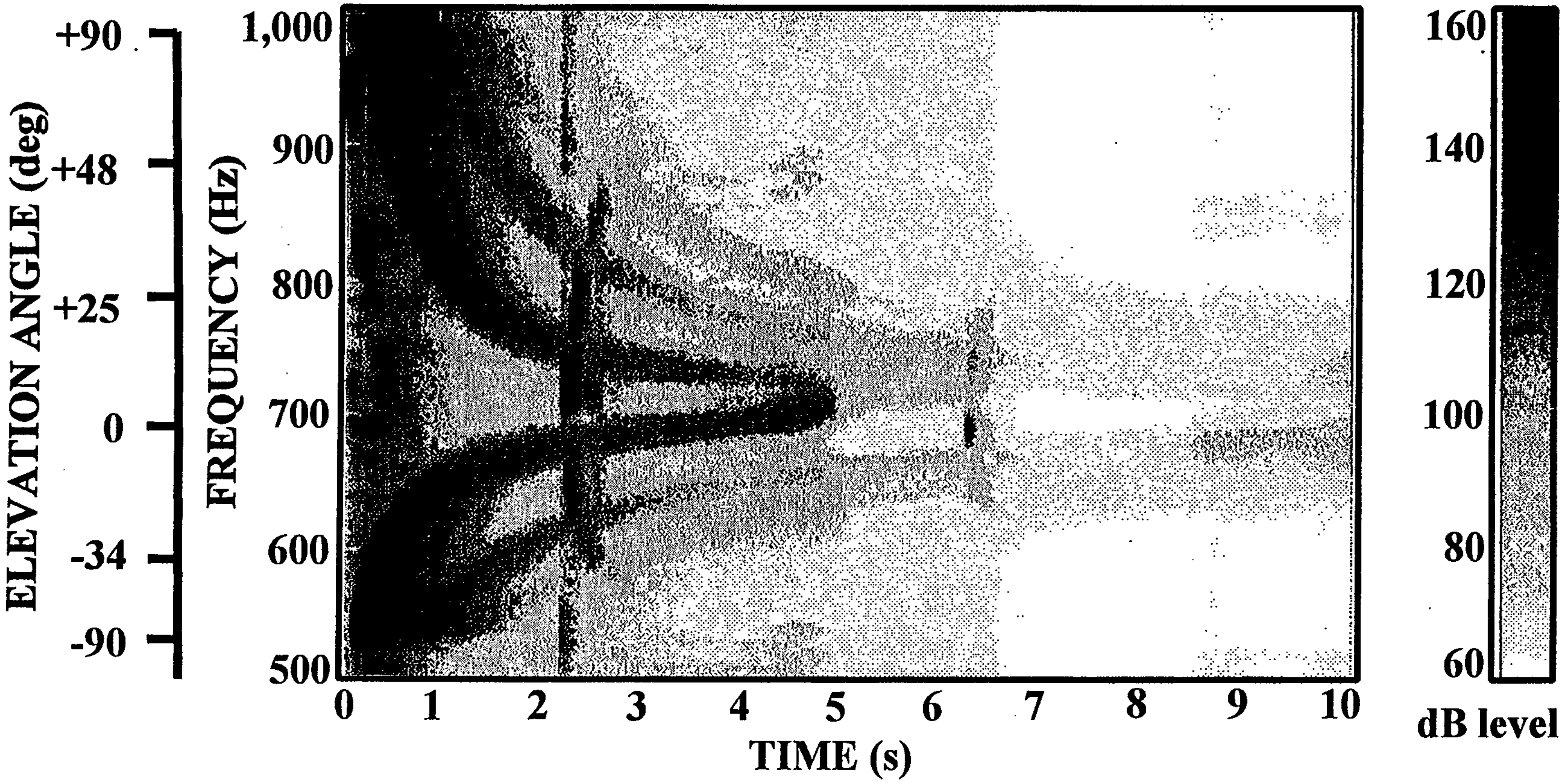

**Figure 27.** Predicted reverberation and echoes on monostatic omnidirectional receiver for conditions of figure 26. Both source and receiver are at a 0 range and a 100-meter depth. Echoes from targets at a 100-meter depth are shown at 2.2 and 6.2 seconds, corresponding to ranges of 1 and 2.5 nmi.

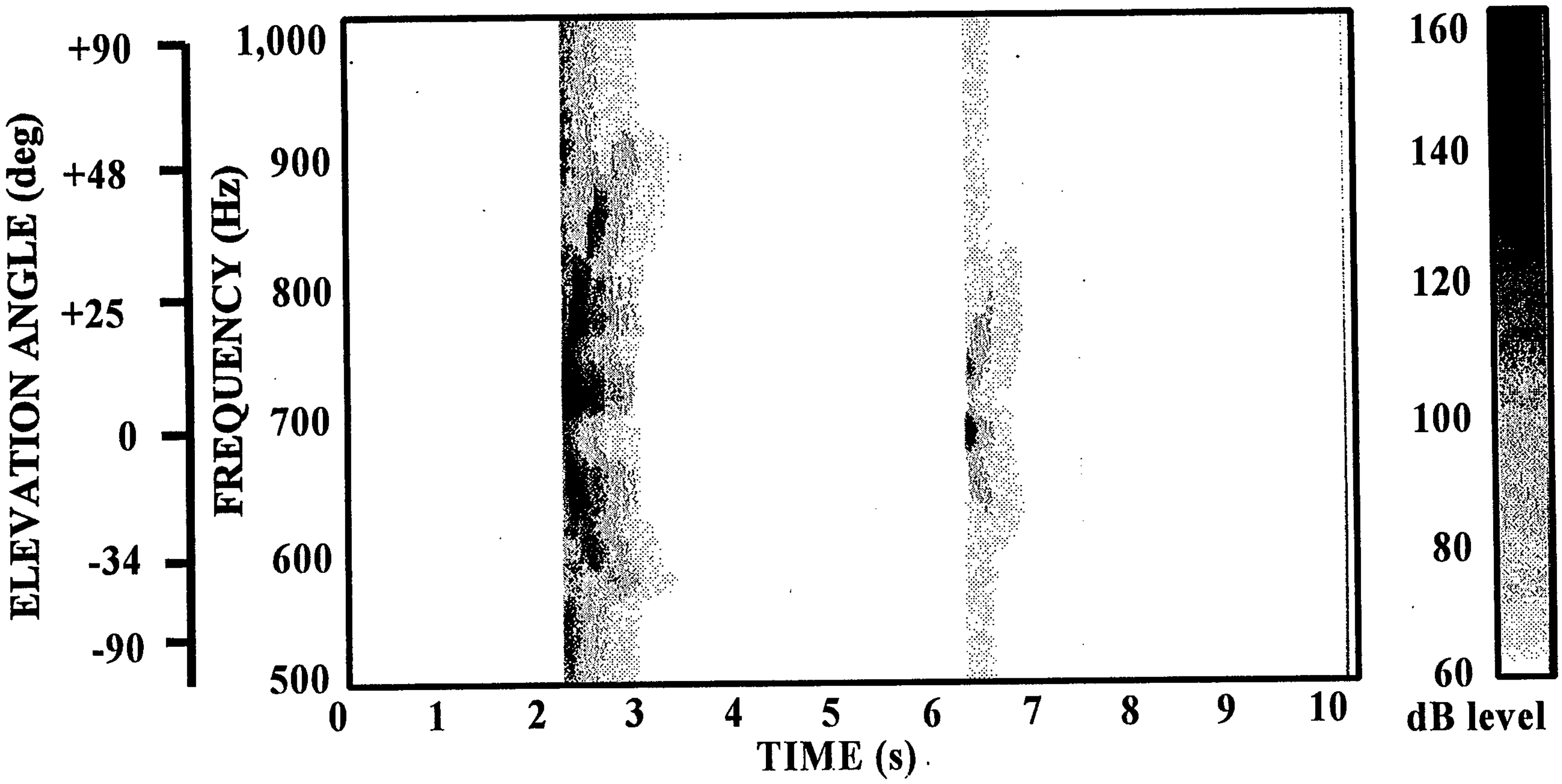

**Figure 28.** Predicted reverberation on omnidirectional receiver for conditions of figure 26. Echoes from 50-dB targets at a 100-meter depth are shown at 2.2 and 6.2 seconds, corresponding to ranges of 1 and 2.5 nmi.

Only the echoes are shown in figure 28. The concave shape of the echoes on different beams (frequencies) reflects the additional delay encountered in the set of paths contributing to the echo. The echo around 714 Hz and 2.2 seconds is the direct path to the target. The echoes at 800 Hz and 650 Hz and 2.3 seconds are bottom-bounce paths to the target. Multiple echoes at a given frequency result from multiple paths back to the receiver, which is colocated with the source in this example. The vertical lines associated with each target are sidelobes of the main detection. Since the sidelobe levels are approximately –30 dB, their amplitude reflects what would be achieved with a 20-dB target strength.

Since echo-to-reverberation ratio is independent of energy source level, the effect of the lower SFS ESL is to reduce the range at which detection becomes noise-limited. The multiple beams provide coverage at all ranges. Target detections are made on beams corresponding to several bottom and surface interacting paths, providing a template for clutter rejection and target depth estimation.

# SUMMARY AND RECOMMENDATIONS

The multimode acoustic source is a high-energy deployable active sonar source which can be programmed remotely to ping in different modes at selectable depths. Each mode has a different combination of spectra and beam patterns. The combination of depths and modes coupled with deployed receivers provides additional capability to optimize the performance of multistatic airborne active fields in various water depths, acoustic environments, and tactical scenarios. For noise-limited applications, high-energy pulses are focused on acoustic paths which propagate to target ranges and depths while minimizing the energy on paths that are absorbed or generate reverberation. In reverberation-limited environments, multiple beams at different frequencies are generated to resolve the target from the reverberation.

It is recommended that a multimode acoustic source be developed. Such a development would demonstrate electronic timing and firing technology, first with circuit board and discrete component electronics and then with hybrid integrated electronic timing and firing circuitry. The critical component is the electronic initiator (EI) circuit. One EI is required for each source array element. Approximately 48 EI circuits would be used per array, with a few hundred per deployed source. A development program would focus on achieving an EI which can be fabricated in large quantities at $1 to $2 each. If this can be achieved, expandable multimode sources with four pings per A-size source would be able to be fabricated for less than $1,500.

Additional analyses are also recommended to determine the echo-to-reverberation and echo-to-noise performance of candidate source mode–receiver systems and geometries in sloping environments that are typical of the transitional waters between shallow waters and deep noise-limited waters.

# APPENDIX A

## SOURCE LEVELS OF EXPLOSIVE SOURCES

The pressure at range *r* vs time from an SUS shock wave is given by[6]

$$p(t) \;=\; p_o e^{-t/t_0} \tag{A-1}$$

$$p_o \;=\; 2.16 \;x\; 10^4 \left(\frac{w^{1/3}}{r}\right)^{1.13} \tag{A-2}$$

$$t_o \;=\; 58\, w^{1/3} \left(\frac{w^{1/3}}{r}\right)^{-0.22} \tag{A-3}$$

where $p_o$ is in lb/in$^2$, $w$ is in pounds, $r$ is in feet, and $t_o$ is in µs.

Changing units

$$p_o = 2.16 \times 10^4 \left(\frac{w(\text{lb})^{1/3}}{r(\text{m}) \times 3.28 \text{ ft/m}}\right)^{1.13} \text{lb/in}^2$$

$$\times \frac{0.4536 \text{ kgms/lb} \times 9.8 \text{ m}^2}{(0.0254)^2\text{m}^2/\text{in}^2} \times \frac{1\mu\text{Pa}}{10^{-6}\text{Newton/m}^2} \tag{A-4}$$

$$= 3.89 \times 10^{13} \left(\frac{w^{1/3}}{r}\right)^{1.13} \mu\text{Pa}$$

and

$$t_o = 58\; w^{1/3} \left(\frac{w^{1/3}}{r(\text{m}) \times 3.28 \text{ ft/m}}\right)^{-0.22} \mu\text{s}$$

$$= 75.33\; w^{0.26} r^{0.22} \mu\text{s} \tag{A-5}$$

where $r$ is in meters and $w$ is in pounds.

Let's choose 100 meters as a point to evaluate $p_o$ and $t_o$ which is a typical range at which the measurements were made.[7] We will also propagate $p_o$ back to the reference range of 1 meter, assuming propagation as $p \;\propto\; 1/r$ and energy $\;\propto\; 1/r^2$.

Thus

[6] Urick, *op cit.*

[7] The choice of evaluation range is somewhat arbitrary. It is chosen greater than 1 meter to account for the difference between the nonlinear propagation of Eq. A-2 and the linear propagation assumed in acoustic propagation models. The difference between choosing 10 meters or 1,000 meters ranter than 100 meters as the evaluation point is ± 1.8 dB.

$$p_o = 3.89 \times 10^{13} \left(\frac{w^{1/3}}{100}\right)^{1.13} \times 100$$

(A-6)

$$= 2.14 \times 10^{13} (w)^{0.377} \mu\text{P at 1 m}$$

and

$$t_o = 75.33\ w^{0.26} \times 100^{0.22}$$

(A-7)

$$= 207\ w^{0.26} \mu\text{s}$$

## APPENDIX A

## SOURCE CODE:

CPNN.CPP
START.CPP
SOIL.C

```
//                            CPNN.CPP
//              Program to Generate Data for Input for CPTINT

//                 by: John Andrews,  12 August 1993
//                    Computer Sciences Corporation
//
//
//  This program creates artificial input data for "CPTINT".  "CPTINT" is
//  a cone data interpretation program distributed by the Civil Engineering
//  Department, University of British Colombia, Vancouver, B.C. Canada
//  phone number: (604) 822-2637.
//
//  Execution of this program creates a file named "nncpt.dat". It contains
//  three space delimited fields, Depth, Qc, and Fs.  A new line separates
//  records. The file "nncpt.dat" can be read directly by "CPTINT".
//
//
// -------------------------------------------------------------------------


   #include <iostream.h>
   #include <fstream.h>
   #include <math.h>

   main()
   {

   float qc;                                   // cone bearing (bar)
   int depth = 0;                              // dummy depth variable

   ofstream nncpt("nncpt.dat");                // create file nncpt.dat

    for(float x=0; x<=3; x+=0.03){             // divide y axis into 101 pixels
     qc = pow(10.0,x);                         // compute qc
      for (int n=1; n<=120; n++){              // divide x axis into 120 pixels
       nncpt << depth++ << "  " ;              // writes & increments depth
       nncpt << qc << "  " ;                   // write qc to file
       nncpt << qc*(.08*n/120) << '\n';        // write fs to file
      }
    }
   }
```

```
//                         START.CPP
//               by: John Andrews, August 1993
//               Computer Sciences Corporation
//
//
//   This program reads a NeuralWare *.nnr file having 2 input fields
//   and 12 output fields (each desired and actual).  The input file
//   is entered as a command line argument.  The program creates two files
//   for storing correctly and incorrectly classified records.  It also
//   displays the number of correct and incorrect classifications.
//
//  ------------------------------------------------------------------------


   #include <iostream.h>
   #include <fstream.h>
   #include <math.h>
   #include <graphics.h>


   main(int argc, char *argv[])
   {
     float input1, input2, maxvalue;
     float desout[12];
     float actout[12];
     int i, desclass, actclass;
     int numcor = 0, nummis = 0;

     cout << argv[1];

     ifstream nnrfile(argv[1]);             //input file
     if (!nnrfile) {                        //error handling
     cerr << "Can't open the file";
     return(0);
     }

     ofstream corfile("cor.nnr");           //create files for correct
     ofstream misfile("mis.nnr");           //& incorrect records

     while (!nnrfile.eof())  {              //read 2 input fields
       nnrfile >> input1;
       nnrfile >> input2;

       for(i=0; i<12; i++){                 //read 12 desired output fields
           nnrfile >> desout[i];
           if (desout[i] == 1.00)           //determine desired class
             desclass = i;
       }

       nnrfile >> actout[0];                //read 12 actual output fields
       actclass = 0;
       maxvalue = actout[0];

       for(i=1;  i<12; i++){                //determine highest actual output
           nnrfile >> actout[i];
           if( actout[i] > maxvalue ){
              actclass = i;
              maxvalue = actout[i];
           }
```

```
    }
                                      //write correctly classified
                                      //records to file
    if (actclass == desclass) {
       numcor++;
       corfile << input1 << "\t" << input2 << "\t" << actclass + 1 <<"\n";
       }
                                      //write incorrectly classified
                                      //records to other file
    else{
       nummis++;
       misfile << input1 << "\t" << input2 << "\t" << actclass + 1 << "\n";
       }


                                   //output results of last
                                   //record (for code testing)

  for(i=0;i<12;i++)
  cout<< desout[i] << "   "  << actout[i] << "\n";
  cout<< desclass << "   "  << actclass << "   " << maxvalue <<"\n\n\n";
  cout << input1 << "   " << input2 << "\n";
  cout << numcor << "   " << nummis << "\n\n"; //output numbers cor & !cor

}
```

```
/*                              SOIL.C                                          */
/*                                                                              */
/*                 by: John Andrews,  September 1993                            */
/*                   Computer Sciences Corporation                              */

/*     This program implements a four layer feedforward neural network         */
/*     for CPT soil classification.                                             */
/*     It takes two parameters, log Qc and Rf (TSF units).                      */
/*     It returns soil class integer 1-12.                                      */
/*     The network has a 2x15x15x12 architecture.                               */
/*     The output layer is competitive (recall mode).                           */
/*                                                                              */
/*     Mon Sep 20 14:36:49 1993 (cp15.c)                                        */
/*     Recall-Only Run-time for <cp15x15b>                                      */
/*     Modification of NeuralWorks Flash code.                                  */
/*     Control Strategy is: <backprop>                                          */
/*    ----------------------------------------------------------------------    */

  #include <math.h>

  int soil( float cone, float csratio )  /* log cone pressure, sleeve/cone *
  {
    float  Xout[46];   /* work arrays */
    float  CmpV;       /* competitive value */
    int    ICmpX;      /* compeititive index */


    /* Read and scale input into network */
    Xout[2] = cone * (0.67342337) + (-1.0328631);
    Xout[3] = csratio * (0.25211143) + (-1.0168915);

    /* Generating code for PE 0 in layer 3 */
    Xout[4]  = (float)(-1.3774354) + (float)(4.3437333) * Xout[2] +
        (float)(0.70192885) * Xout[3];
    Xout[4] = tanh( Xout[4] );

    /* Generating code for PE 1 in layer 3 */
    Xout[5]  = (float)(-0.67682087) + (float)(-6.8595533) * Xout[2] +
        (float)(1.1557226) * Xout[3];
    Xout[5] = tanh( Xout[5] );

    /* Generating code for PE 2 in layer 3 */
    Xout[6]  = (float)(0.96711385) + (float)(-2.6023602) * Xout[2] +
        (float)(0.2549077) * Xout[3];
    Xout[6] = tanh( Xout[6] );

    /* Generating code for PE 3 in layer 3 */
    Xout[7]  = (float)(-1.7575909) + (float)(-8.4847059) * Xout[2] +
        (float)(2.479882) * Xout[3];
    Xout[7] = tanh( Xout[7] );

    /* Generating code for PE 4 in layer 3 */
    Xout[8]  = (float)(5.9350553) + (float)(6.4196491) * Xout[2] +
        (float)(3.760468) * Xout[3];
    Xout[8] = tanh( Xout[8] );

    /* Generating code for PE 5 in layer 3 */
    Xout[9]  = (float)(0.92435199) + (float)(0.45118952) * Xout[2] +
        (float)(-8.390686) * Xout[3];
```

```
Xout[9] = tanh( Xout[9] );

/* Generating code for PE 6 in layer 3 */
Xout[10]  = (float)(-4.9638495) + (float)(9.5670557) * Xout[2] +
    (float)(3.2453654) * Xout[3];
Xout[10] = tanh( Xout[10] );

/* Generating code for PE 7 in layer 3 */
Xout[11]  = (float)(-3.380995) + (float)(2.4127712) * Xout[2] +
    (float)(-8.394412) * Xout[3];
Xout[11] = tanh( Xout[11] );

/* Generating code for PE 8 in layer 3 */
Xout[12]  = (float)(-10.241221) + (float)(-12.071634) * Xout[2] +
    (float)(4.0742674) * Xout[3];
Xout[12] = tanh( Xout[12] );

/* Generating code for PE 9 in layer 3 */
Xout[13]  = (float)(6.7051864) + (float)(-1.3762254) * Xout[2] +
    (float)(7.9877915) * Xout[3];
Xout[13] = tanh( Xout[13] );

/* Generating code for PE 10 in layer 3 */
Xout[14]  = (float)(-2.9832273) + (float)(-0.30514637) * Xout[2] +
    (float)(-6.6823497) * Xout[3];
Xout[14] = tanh( Xout[14] );

/* Generating code for PE 11 in layer 3 */
Xout[15]  = (float)(1.1439002) + (float)(-3.3561134) * Xout[2] +
    (float)(11.728964) * Xout[3];
Xout[15] = tanh( Xout[15] );

/* Generating code for PE 12 in layer 3 */
Xout[16]  = (float)(0.6612637) + (float)(-6.8429189) * Xout[2] +
    (float)(0.44200832) * Xout[3];
Xout[16] = tanh( Xout[16] );

/* Generating code for PE 13 in layer 3 */
Xout[17]  = (float)(3.5646832) + (float)(0.60023791) * Xout[2] +
    (float)(6.4878182) * Xout[3];
Xout[17] = tanh( Xout[17] );

/* Generating code for PE 14 in layer 3 */
Xout[18]  = (float)(0.9410491) + (float)(-4.5457315) * Xout[2] +
    (float)(-0.30045751) * Xout[3];
Xout[18] = tanh( Xout[18] );

/* Generating code for PE 0 in layer 4 */
Xout[19]  = (float)(-0.34009913) + (float)(0.6707626) * Xout[4] +
    (float)(-1.4701272) * Xout[5] + (float)(-0.70717514) * Xout[6] +
    (float)(-1.9541312) * Xout[7] + (float)(3.078223) * Xout[8] +
    (float)(2.21312) * Xout[9] + (float)(-1.4408562) * Xout[10] +
    (float)(-1.6046822) * Xout[11] + (float)(-5.0437288) * Xout[12] +
    (float)(1.2409499) * Xout[13] + (float)(-0.78926444) * Xout[14] +
    (float)(2.8454196) * Xout[15] + (float)(-0.73372024) * Xout[16] +
    (float)(0.87041885) * Xout[17] + (float)(-0.75311923) * Xout[18];
Xout[19] = tanh( Xout[19] );

/* Generating code for PE 1 in layer 4 */
Xout[20]  = (float)(-2.8791444) + (float)(0.48050028) * Xout[4] +
    (float)(-2.9404998) * Xout[5] + (float)(-0.64095086) * Xout[6] +
    (float)(-2.8091729) * Xout[7] + (float)(0.51963592) * Xout[8] +
    (float)(1.7173361) * Xout[9] + (float)(1.9665118) * Xout[10] +
    (float)(3.4087687) * Xout[11] + (float)(2.4434633) * Xout[12] +
    (float)(-2.1504614) * Xout[13] + (float)(-0.27361864) * Xout[14] +
    (float)(-5.3734245) * Xout[15] + (float)(-0.30357832) * Xout[16] +
    (float)(0.31644058) * Xout[17] + (float)(-0.16554987) * Xout[18];
Xout[20] = tanh( Xout[20] );

/* Generating code for PE 2 in layer 4 */
Xout[21]  = (float)(-1.1246655) + (float)(1.6832063) * Xout[4] +
    (float)(-1.8406967) * Xout[5] + (float)(-1.0049504) * Xout[6] +
    (float)(-0.74303836) * Xout[7] + (float)(1.5295234) * Xout[8] +
    (float)(-1.6239283) * Xout[9] + (float)(2.3027546) * Xout[10] +
    (float)(-2.4416902) * Xout[11] + (float)(-2.1821213) * Xout[12] +
    (float)(-3.2663224) * Xout[13] + (float)(2.1976848) * Xout[14] +
    (float)(1.3920369) * Xout[15] + (float)(-3.0185628) * Xout[16] +
    (float)(-2.4525895) * Xout[17] + (float)(-1.8309394) * Xout[18];
Xout[21] = tanh( Xout[21] );

/* Generating code for PE 3 in layer 4 */
Xout[22]  = (float)(0.96787357) + (float)(-0.88769686) * Xout[4] +
    (float)(1.6487899) * Xout[5] + (float)(0.72958428) * Xout[6] +
    (float)(0.33483446) * Xout[7] + (float)(0.12007745) * Xout[8] +
    (float)(-0.61698985) * Xout[9] + (float)(-1.3958707) * Xout[10] +
    (float)(-1.4732052) * Xout[11] + (float)(-1.1600244) * Xout[12] +
    (float)(0.91249472) * Xout[13] + (float)(-1.8335599) * Xout[14] +
    (float)(1.4040639) * Xout[15] + (float)(2.0465801) * Xout[16] +
    (float)(1.3337294) * Xout[17] + (float)(0.92064232) * Xout[18];
Xout[22] = tanh( Xout[22] );

/* Generating code for PE 4 in layer 4 */
Xout[23]  = (float)(-0.36530879) + (float)(-0.59072089) * Xout[4] +
    (float)(1.2441344) * Xout[5] + (float)(0.21524772) * Xout[6] +
    (float)(1.6768456) * Xout[7] + (float)(-0.50899112) * Xout[8] +
    (float)(0.83034575) * Xout[9] + (float)(-0.94205385) * Xout[10] +
    (float)(-1.5979517) * Xout[11] + (float)(0.24534306) * Xout[12] +
    (float)(0.64400119) * Xout[13] + (float)(-0.18732649) * Xout[14] +
    (float)(0.71088886) * Xout[15] + (float)(0.28635231) * Xout[16] +
    (float)(0.10100901) * Xout[17] + (float)(0.48917824) * Xout[18];
Xout[23] = tanh( Xout[23] );

/* Generating code for PE 5 in layer 4 */
Xout[24]  = (float)(-2.2106702) + (float)(-2.2451408) * Xout[4] +
    (float)(1.1351285) * Xout[5] + (float)(1.4046094) * Xout[6] +
    (float)(1.2378136) * Xout[7] + (float)(0.25965616) * Xout[8] +
    (float)(1.9334576) * Xout[9] + (float)(-1.2937686) * Xout[10] +
    (float)(2.5316298) * Xout[11] + (float)(-2.1072848) * Xout[12] +
    (float)(3.8545687) * Xout[13] + (float)(-2.0066607) * Xout[14] +
    (float)(-3.6767368) * Xout[15] + (float)(1.9547799) * Xout[16] +
    (float)(1.8855346) * Xout[17] + (float)(2.3863194) * Xout[18];
Xout[24] = tanh( Xout[24] );

/* Generating code for PE 6 in layer 4 */
Xout[25]  = (float)(1.4030933) + (float)(0.59960848) * Xout[4] +
    (float)(-1.8639184) * Xout[5] + (float)(-0.3860828) * Xout[6] +
    (float)(-4.0644178) * Xout[7] + (float)(3.9791329) * Xout[8] +
    (float)(6.2324586) * Xout[9] + (float)(3.5871241) * Xout[10] +
    (float)(4.2374911) * Xout[11] + (float)(-3.7616525) * Xout[12] +
    (float)(1.1509675) * Xout[13] + (float)(0.32016569) * Xout[14] +
    (float)(-0.77581334) * Xout[15] + (float)(-1.4044687) * Xout[16] +
    (float)(-0.62049145) * Xout[17] + (float)(-0.57259399) * Xout[18];
Xout[25] = tanh( Xout[25] );

/* Generating code for PE 7 in layer 4 */
Xout[26]  = (float)(-0.52494174) + (float)(0.5791747) * Xout[4] +
    (float)(-1.5759159) * Xout[5] + (float)(-0.22171608) * Xout[6] +
    (float)(-3.5944927) * Xout[7] + (float)(-0.24836895) * Xout[8] +
    (float)(2.8628707) * Xout[9] + (float)(-1.4358618) * Xout[10] +
    (float)(1.8196349) * Xout[11] + (float)(0.75441694) * Xout[12] +
    (float)(-0.10356036) * Xout[13] + (float)(-0.0011890249) * Xout[14] +
    (float)(-3.3494186) * Xout[15] + (float)(-1.1160185) * Xout[16] +
    (float)(-0.32207805) * Xout[17] + (float)(-0.64801431) * Xout[18];
Xout[26] = tanh( Xout[26] );

/* Generating code for PE 8 in layer 4 */
Xout[27]  = (float)(1.6821843) + (float)(-1.1902723) * Xout[4] +
    (float)(0.99234951) * Xout[5] + (float)(0.76882249) * Xout[6] +
    (float)(0.4855696) * Xout[7] + (float)(1.5828314) * Xout[8] +
    (float)(1.4035158) * Xout[9] + (float)(-4.6815071) * Xout[10] +
    (float)(2.8950787) * Xout[11] + (float)(-5.4909062) * Xout[12] +
    (float)(-2.4956195) * Xout[13] + (float)(2.2027419) * Xout[14] +
    (float)(-0.70475209) * Xout[15] + (float)(2.4308791) * Xout[16] +
    (float)(-2.0018756) * Xout[17] + (float)(1.2343072) * Xout[18];
Xout[27] = tanh( Xout[27] );

/* Generating code for PE 9 in layer 4 */
Xout[28]  = (float)(0.066524662) + (float)(0.2381712) * Xout[4] +
    (float)(0.36230001) * Xout[5] + (float)(-0.32556629) * Xout[6] +
    (float)(0.84901702) * Xout[7] + (float)(-0.0059411367) * Xout[8] +
    (float)(0.22432239) * Xout[9] + (float)(0.20808095) * Xout[10] +
    (float)(-0.76741439) * Xout[11] + (float)(-1.933818) * Xout[12] +
    (float)(-0.28691539) * Xout[13] + (float)(-0.048413601) * Xout[14] +
    (float)(0.45335075) * Xout[15] + (float)(0.39268398) * Xout[16] +
    (float)(-0.35602769) * Xout[17] + (float)(-0.26055622) * Xout[18];
Xout[28] = tanh( Xout[28] );

/* Generating code for PE 10 in layer 4 */
Xout[29]  = (float)(0.20022397) + (float)(-0.35824397) * Xout[4] +
    (float)(-0.10193466) * Xout[5] + (float)(0.44272116) * Xout[6] +
    (float)(-0.25799647) * Xout[7] + (float)(1.3102238) * Xout[8] +
    (float)(-0.054711644) * Xout[9] + (float)(-1.0969514) * Xout[10] +
    (float)(0.17795166) * Xout[11] + (float)(-0.57731533) * Xout[12] +
    (float)(-0.69488567) * Xout[13] + (float)(1.3976555) * Xout[14] +
    (float)(-1.0415714) * Xout[15] + (float)(0.32586724) * Xout[16] +
    (float)(-1.3062199) * Xout[17] + (float)(0.31708151) * Xout[18];
Xout[29] = tanh( Xout[29] );

/* Generating code for PE 11 in layer 4 */
Xout[30]  = (float)(0.91501057) + (float)(0.32980675) * Xout[4] +
    (float)(-0.065754704) * Xout[5] + (float)(0.15424739) * Xout[6] +
    (float)(-0.2029167) * Xout[7] + (float)(-0.37174666) * Xout[8] +
    (float)(0.082694367) * Xout[9] + (float)(0.036902208) * Xout[10] +
    (float)(-0.23626976) * Xout[11] + (float)(3.6132705) * Xout[12] +
    (float)(0.015108704) * Xout[13] + (float)(0.85283083) * Xout[14] +
    (float)(-0.93645924) * Xout[15] + (float)(-0.19907707) * Xout[16] +
    (float)(-0.34797958) * Xout[17] + (float)(-0.17012013) * Xout[18];
Xout[30] = tanh( Xout[30] );

/* Generating code for PE 12 in layer 4 */
```

```
Xout[31]  = (float)(-0.19556384) + (float)(-1.4724393) * Xout[4] +
    (float)(-1.9010726) * Xout[5] + (float)(1.1318318) * Xout[6] +
    (float)(-3.6829324) * Xout[7] + (float)(4.0976496) * Xout[8] +
    (float)(-0.15674546) * Xout[9] + (float)(-0.92516965) * Xout[10] +
    (float)(-0.050786) * Xout[11] + (float)(-1.3713385) * Xout[12] +
    (float)(4.3936687) * Xout[13] + (float)(-2.2437789) * Xout[14] +
    (float)(0.71045846) * Xout[15] + (float)(-0.16340141) * Xout[16] +
    (float)(2.8538706) * Xout[17] + (float)(1.0720434) * Xout[18];
Xout[31] = tanh( Xout[31] );

/* Generating code for PE 13 in layer 4 */
Xout[32]  = (float)(0.79607087) + (float)(1.4678394) * Xout[4] +
    (float)(-3.0694056) * Xout[5] + (float)(-0.62680197) * Xout[6] +
    (float)(-1.2404653) * Xout[7] + (float)(1.1638988) * Xout[8] +
    (float)(-1.7049054) * Xout[9] + (float)(2.3438494) * Xout[10] +
    (float)(1.0563062) * Xout[11] + (float)(-1.2724484) * Xout[12] +
    (float)(-1.7043641) * Xout[13] + (float)(3.1809888) * Xout[14] +
    (float)(1.8637894) * Xout[15] + (float)(-3.2304156) * Xout[16] +
    (float)(-2.3590434) * Xout[17] + (float)(-1.7398988) * Xout[18];
Xout[32] = tanh( Xout[32] );

/* Generating code for PE 14 in layer 4 */
Xout[33]  = (float)(1.5249654) + (float)(-0.061132118) * Xout[4] +
    (float)(1.0983677) * Xout[5] + (float)(0.44974536) * Xout[6] +
    (float)(2.3553016) * Xout[7] + (float)(-1.6995065) * Xout[8] +
    (float)(-2.5550337) * Xout[9] + (float)(2.0845053) * Xout[10] +
    (float)(-0.31993854) * Xout[11] + (float)(-2.2329035) * Xout[12] +
    (float)(-0.79235321) * Xout[13] + (float)(0.25259238) * Xout[14] +
    (float)(1.8125871) * Xout[15] + (float)(-0.27512208) * Xout[16] +
    (float)(-0.17584039) * Xout[17] + (float)(-0.065280482) * Xout[18];
Xout[33] = tanh( Xout[33] );

/* Generating code for PE 0 in layer 5 */
Xout[34]  = (float)(-0.87790149) + (float)(-1.0692201) * Xout[19] +
    (float)(0.03755356) * Xout[20] + (float)(0.096775234) * Xout[21] +
    (float)(-0.049250424) * Xout[22] + (float)(0.10390767) * Xout[23] +
    (float)(-0.096084803) * Xout[24] + (float)(0.014091688) * Xout[25] +
    (float)(0.0051135253) * Xout[26] + (float)(0.06910155) * Xout[27] +
    (float)(-0.041864727) * Xout[28] + (float)(0.064484537) * Xout[29] +
    (float)(-1.089339) * Xout[30] + (float)(-0.084535331) * Xout[31] +
    (float)(-0.016315376) * Xout[32] + (float)(0.021016486) * Xout[33];
Xout[34] = tanh( Xout[34] );
CmpV = Xout[34]; ICmpX = 34; /* start competition */

/* Generating code for PE 1 in layer 5 */
Xout[35]  = (float)(-0.23333962) + (float)(-0.23561044) * Xout[19] +
    (float)(-0.050891768) * Xout[20] + (float)(-0.10643666) * Xout[21] +
    (float)(0.018222565) * Xout[22] + (float)(-0.097888283) * Xout[23] +
    (float)(0.047541238) * Xout[24] + (float)(-0.059924994) * Xout[25] +
    (float)(-0.028140228) * Xout[26] + (float)(-0.12535518) * Xout[27] +
    (float)(0.040437236) * Xout[28] + (float)(-0.0028511814) * Xout[29] +
    (float)(0.92822611) * Xout[30] + (float)(0.21035294) * Xout[31] +
    (float)(0.007625306) * Xout[32] + (float)(-0.068101816) * Xout[33];
Xout[35] = tanh( Xout[35] );
if ( Xout[35] > CmpV ) { CmpV = Xout[35]; ICmpX = 35; } /* track winner */

/* Generating code for PE 2 in layer 5 */
Xout[36]  = (float)(-2.3662107) + (float)(0.93160915) * Xout[19] +
    (float)(-0.11621797) * Xout[20] + (float)(0.024368644) * Xout[21] +
    (float)(-0.14027528) * Xout[22] + (float)(-0.2152873) * Xout[23] +
    (float)(0.01436625) * Xout[24] + (float)(-1.1697067) * Xout[25] +
    (float)(0.037573472) * Xout[26] + (float)(0.011277422) * Xout[27] +
    (float)(0.094070829) * Xout[28] + (float)(0.05417366) * Xout[29] +
    (float)(-0.16597992) * Xout[30] + (float)(1.2443105) * Xout[31] +
    (float)(-0.14807294) * Xout[32] + (float)(0.020404585) * Xout[33];
Xout[36] = tanh( Xout[36] );
if ( Xout[36] > CmpV ) { CmpV = Xout[36]; ICmpX = 36; } /* track winner */

/* Generating code for PE 3 in layer 5 */
Xout[37]  = (float)(-1.1655524) + (float)(0.14780924) * Xout[19] +
    (float)(0.20306671) * Xout[20] + (float)(-0.051036075) * Xout[21] +
    (float)(-0.59072953) * Xout[22] + (float)(0.22242334) * Xout[23] +
    (float)(0.1092864) * Xout[24] + (float)(0.98674947) * Xout[25] +
    (float)(-1.0761287) * Xout[26] + (float)(0.11027345) * Xout[27] +
    (float)(0.086110197) * Xout[28] + (float)(-0.13758875) * Xout[29] +
    (float)(0.29757303) * Xout[30] + (float)(-0.018368572) * Xout[31] +
    (float)(-0.53483301) * Xout[32] + (float)(-0.22306475) * Xout[33];
Xout[37] = tanh( Xout[37] );
if ( Xout[37] > CmpV ) { CmpV = Xout[37]; ICmpX = 37; } /* track winner */

/* Generating code for PE 4 in layer 5 */
Xout[38]  = (float)(-2.9970107) + (float)(1.1596954) * Xout[19] +
    (float)(-1.1503054) * Xout[20] + (float)(0.005177083) * Xout[21] +
    (float)(-0.096239172) * Xout[22] + (float)(-0.024488389) * Xout[23] +
    (float)(-0.051597442) * Xout[24] + (float)(0.28520513) * Xout[25] +
    (float)(1.0015672) * Xout[26] + (float)(0.81431431) * Xout[27] +
    (float)(-0.1415341) * Xout[28] + (float)(0.026307182) * Xout[29] +
    (float)(-0.053032927) * Xout[30] + (float)(-0.011248631) * Xout[31] +
    (float)(-0.10334851) * Xout[32] + (float)(0.28688434) * Xout[33];
Xout[38] = tanh( Xout[38] );
if ( Xout[38] > CmpV ) { CmpV = Xout[38]; ICmpX = 38; } /* track winner */

/* Generating code for PE 5 in layer 5 */
Xout[39]  = (float)(-1.6181245) + (float)(0.03928183) * Xout[19] +
    (float)(0.9619233) * Xout[20] + (float)(0.0081323469) * Xout[21] +
    (float)(0.34299639) * Xout[22] + (float)(-0.15945068) * Xout[23] +
    (float)(-0.1256876) * Xout[24] + (float)(0.59323692) * Xout[25] +
    (float)(0.030090949) * Xout[26] + (float)(0.044754967) * Xout[27] +
    (float)(-0.067690797) * Xout[28] + (float)(0.001090379) * Xout[29] +
    (float)(0.016745619) * Xout[30] + (float)(0.56472063) * Xout[31] +
    (float)(-0.83290011) * Xout[32] + (float)(0.6618259) * Xout[33];
Xout[39] = tanh( Xout[39] );
if ( Xout[39] > CmpV ) { CmpV = Xout[39]; ICmpX = 39; } /* track winner */

/* Generating code for PE 6 in layer 5 */
Xout[40]  = (float)(-1.0943762) + (float)(-0.035113763) * Xout[19] +
    (float)(-0.24052712) * Xout[20] + (float)(-1.1928893) * Xout[21] +
    (float)(-0.30317307) * Xout[22] + (float)(-0.15959917) * Xout[23] +
    (float)(-0.07071659) * Xout[24] + (float)(0.0035665662) * Xout[25] +
    (float)(-0.014620778) * Xout[26] + (float)(-0.01583972) * Xout[27] +
    (float)(-0.052799445) * Xout[28] + (float)(0.0028659943) * Xout[29] +
    (float)(-0.098308317) * Xout[30] + (float)(0.010499664) * Xout[31] +
    (float)(0.93895841) * Xout[32] + (float)(0.01171761) * Xout[33];
Xout[40] = tanh( Xout[40] );
if ( Xout[40] > CmpV ) { CmpV = Xout[40]; ICmpX = 40; } /* track winner */

/* Generating code for PE 7 in layer 5 */
Xout[41]  = (float)(-1.528774) + (float)(0.044951025) * Xout[19] +
    (float)(-0.24355549) * Xout[20] + (float)(1.1148211) * Xout[21] +
    (float)(0.20645723) * Xout[22] + (float)(-0.12182666) * Xout[23] +
```

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*

| | | |
|---|---|---|
| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE<br>December 1999 | 3. REPORT TYPE AND DATES COVERED<br>Final: Jan 1993 – Sep 1993 |

4. TITLE AND SUBTITLE

MULTIMODE ACOUSTIC SOURCE

5. FUNDING NUMBERS

N2269193/WY00199
A5465461/003C/3H2089001

6, AUTHOR(S)

Newell O. Booth and Shelby F. Sullivan, Sr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

SSC San Diego
San Diego, CA 92152–5001

8. PERFORMING ORGANIZATION REPORT NUMBER

TR 1812

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Air Warfare Center
Code 16
Warminster, PA 18974–0591

Naval Air Systems Command
AIR–546W8
Arlington, VA 22202

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A deployed impulsive acoustic source can be programmed remotely (as from an aircraft) to ping in different modes at selectable depths, each mode having a different combination of spectra and beampatterns. The choice of modes is achieved by controlling the timing and firing order of explosive elements positioned in the array. Mode frequencies, beamwidths, and associated bandwidths are determined at the time of the array's manufacture by the number of elements and the spacing. Each of the modes has different beam and spectral characteristics which are useful in different environments and in different tactical situations. The acoustic characteristics of the source in each of the modes are described, and acoustic design criteria are derived. Application examples are illustrated.

14. SUBJECT TERMS

Mission Area: Surveillance
acoustic sound sources
acoustic source arrays
airborne active sonars
explosive sound sources
multistatic active sonars

15. NUMBER OF PAGES

68

16. PRICE CODE

| | | | |
|---|---|---|---|
| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | SAME AS REPORT |

| | | |
|---|---|---|
| 21a. NAME OF RESPONSIBLE INDIVIDUAL<br>A. T. Abawi | 21b. TELEPHONE *(include Area Code)*<br>(619) 553–3101<br>e-mail: abawi@spawar.navy.mil | 21c. OFFICE SYMBOL<br>D881 |

# INITIAL DISTRIBUTION

| | | |
|---|---|---|
| D0012 | Patent Counsel | (1) |
| D0271 | Archive/Stock | (6) |
| D0274 | Library | (2) |
| D027 | M. E. Cathcart | (1) |
| D0271 | D. Richter | (1) |
| D881 | A. T. Abawi | (10) |

Defense Technical Information Center
Fort Belvoir, VA 22060–6218 (4)

SSC San Diego Liaison Office
Arlington, VA 22202–4804

Center for Naval Analyses
Alexandria, VA 22302–0268

Navy Acquisition, Research and
Development Information Center
Arlington, VA 22202–3734

Government Industry Data Exchange
Program Operations Center
Corona, CA 91718–8000